Introduzione all'arte della composizione tipografica con LaTeX

GUIT

Versione 0.10 — 2007/08/22

Quest'opera è soggetta alla Creative Commons Public License versione 2.5 o posteriore. L'enunciato integrale della Licenza in versione 2.5 è reperibile all'indirizzo internet http://creativecommons.org/licenses/by-nc-nd/2.5/deed.it.

- Si è liberi di riprodurre, distribuire, comunicare al pubblico, esporre, in pubblico, rappresentare, eseguire e recitare quest'opera alle seguenti condizioni:

  Attribuzione Bisogna attribuire la paternità dell'opera nei modi indicati dall'autore o da colui al quale è stata data quest'opera in licenza; in questo caso si tratta del Gruppo degli Utilizzatori Italiani di TEX, GUIT.

  Non commerciale Non si può usare quest'opera per fini commerciali.

  Non opere derivate Non si può alterare o trasformare quest'opera, né usarla per crearne un'altra.

- Ogni volta che si usa o si distribuisce quest'opera, lo si deve fare secondo i termini di questa licenza, che va comunicata con chiarezza.

- In ogni caso si possono concordare con il titolare dei diritti d'autore (il GUIT, in questo caso) utilizzi di quest'opera non consentiti da questa licenza.

I nomi commerciali, i loghi, i trademark appartengono ai rispettivi proprietari.

La foto 2.2 della pagina 6 e le foto delle pagine 9, 82, e 194 — 196 sono di Claudio Beccari.
L'immagine 2.1 della pagina 6 è stata ricavata da un indirizzo Internet non più reperibile; l'autore sembra essere Hermann Zapf.

Coordinatore: Claudio Beccari

Hanno collaborato a questo testo: Luciano Battaia, Paolo Biffis, Riccardo Campana, Gustavo Cevolani, Maurizio Himmelmann, Jerónimo Leal, Gianluca Pignalberi, Ottavio Rizzo, Lorenzo sit6113, Salvatore Schirone, Andrea Tonelli, Emanuele Z.

Indice

Elenco delle tabelle

Elenco delle figure

Presentazione

Questa guida alla composizione di testi mediante il programma LaTeX è stata predisposta da un gruppo di membri del Gruppo degli Utilizzatori Italiani di TeX, GuIt, il cui obiettivo è proprio quello di far conoscere il sistema di composizione sviluppato dal matematico Donald E. Knuth ormai quasi trenta anni fa. Non si tratta di un programma di interesse archeologico, perché esso è vivo e vegeto, ha dato origine a un buon numero di discendenti e alcune sue parti sono usate all'interno di altri programmi di elaborazione di testi, senza che gli utenti di questi programmi lo sappiano.

Il sistema TeX è stato uno dei primi esempi di software libero; questa sua qualità ne ha permesso il contributo creativo e/o critico di una moltitudine di utenti, come succede sempre con il software libero, per cui si è arricchito nel tempo di una moltitudine di estensioni che gli permettono di comporre praticamente qualsiasi cosa, tranne, forse, certi tipi di depliant pubblicitari.

Si tratta di un programma di composizione tipografica, non di un impaginatore; quest'ultimo tipo di programmi consente di agire sul materiale da impaginare come se fosse una figura da modificare o da adattare, anche se svolge in parte le funzioni di compositore.

LaTeX consente di comporre tipograficamente dei testi contenenti testo corrente, sia in prosa sia in poesia, scritti in qualunque alfabeto, per esempio latino, greco, cirillico, dall'andamento diretto (da sinistra a destra), oppure ebraico, arabo, dall'andamento inverso (da destra a sinistra), oppure cinese, giapponese, coreano, dall'andamento verticale.

Può gestire font di ogni genere, sia di quelli a matrici di punti, sia i font PostScript, sia i font TrueType, OpenType, eccetera. Tali font possono essere codificati in varie maniere, ma alcuni 'figli' di TeX gestiscono anche i font codificati secondo la norma UNICODE.

La caratteristica che più differenzia il sistema TeX dagli altri 'word processor' è il fatto che per comporre un documento con questo sistema bisogna agire in tempi diversi per introdurre il testo e per comporlo; in questo non è molto diverso da certi procedimenti professionali di impaginazione, dove il testo da comporre viene introdotto in un file di solo testo che poi viene in un secondo tempo fatto fluire dentro il programma di impaginazione, assemblandolo insieme alle figure e all'altro materiale non testuale per generare il documento 'finito', pronto da inviare alla fotoincisione e alla stampa.

Invece questo modo di comporre è molto diverso da quello dei word processor, dove il compositore vede direttamente sullo schermo del suo elaboratore il testo già composto, così da poter esaminare immediatamente il frutto del suo lavoro; l'analisi di questa differenza nel modo di procedere verrà svolta nel primo capitolo.

Il secondo capitolo darà al lettore alcune nozioni di tipografia, se non altro per familiarizzarlo con alcune parole che ricorrono spesso nella descrizione delle varie operazioni compositive. Chi avesse già queste nozioni può saltare la lettura di questo capitolo, ma se, nonostante tutto, decidesse di leggerlo, potrebbe constatare che alcune nozioni e/o alcuni vocaboli in questa guida sono usati per indicare cose leggermente diverse da quelle che conosceva.

Il terzo capitolo esaminerà le procedure da seguire per procurarsi il software del sistema TEX e per installare i programmi e i file accessori.

I capitoli quattro, cinque e sei mostrano i primi rudimenti della composizione asincrona mediante il programma LATEX (parte del sistema TEX). Si parlerà essenzialmente di composizione testuale, anche in forma di tabelle.

I capitoli sette e otto si dedicano alla creazione, manipolazione e inclusione di figure.

I capitoli nove e dieci si dedicano invece alla composizione della matematica; se il lettore ha familiarità con manuali di tipografia, avrà notato che l'argomento della composizione della matematica è praticamente assente da quei manuali, perché si tratta di un tipo di composizione abbastanza specializzato e che interviene abbastanza raramente nei libri pubblicati. Per altro un documento non è necessariamente un libro; può essere un rapporto tecnico, una tesi di laurea o di dottorato, un manuale o un prontuario tecnico, un articolo scientifico, insomma, uno scritto dove la matematica compare spesso in forma avanzata.

La composizione di testi di filologia verosimilmente non contiene una sola formula matematica, ma è un tipo di composizione spesso bidimensionale, come la matematica, che richiede di comprendere a fondo i meccanismi compositivi bidimensionali. A questo tipo di composizione verrà dedicata una appendice apposita, non perché la filologia sia meno importante della matematica, ma perché richiede strumenti particolari che, con il sistema TEX, prendono il nome di file di estensione.

La scelta e la manipolazione dei font sarà l'argomento del capitolo undici; i font, o caratteri da stampa, oggi sono sostanzialmente dei disegni da applicare elettronicamente sullo schermo dell'elaboratore o sulla carta; una volta erano degli oggetti ben tangibili, da quando sono nati come caratteri mobili ad opera di Gutenberg, fino ai giorni nostri sotto forma blocchetti di metallo che riportano su una faccia e in rilievo il disegno del segno da inchiostrare e da applicare alla carta. Naturalmente esistono anche altri procedimenti di stampa che non ricorrono ai caratteri metallici appena menzionati, come il rotocalco, l'offset, eccetera, che oggigiorno sono molto più diffusi che non la 'rilievografia' ottenuta usando i caratteri in rilievo appena descritti.

La scelta e la manipolazione dei font per un sistema di composizione tipografica come quello del sistema TEX, richiede azioni particolari che dipendono da una moltitudine di fattori.

Il programma LATEX si distingue da alcuni suoi fratelli, figli del sistema TEX, per consentire al compositore, oltre che allo scrittore, di concentrasi sul messaggio da trasmettere al lettore, invece che sulla forma da dargli, sulla sua estetica. A seconda del documento da comporre lo stile di composizione può essere molto diverso; anche la semplice impaginazione può assumere aspetti grafici diversi; LATEX consente di scegliere in modo globale il tipo di documento e di apportare piccole modifiche stilistiche durante la composizione, senza che il compositore debba preoccuparsi della nerezza dei caratteri, oppure della distanza del numero indicativo di un paragrafo dall'inizio del suo titolino. A tutte queste cose pensa

la scelta iniziale del tipo di documento e alle opzioni specificate. I capitoli dodici e tredici si occupano appunto di queste cose: il capitolo dodici espressamente per le presentazioni proiettate, e il capitolo tredici per i documenti scritti.

Passando ad argomenti più specializzati, i capitoli quattordici, quindici e sedici si occupano della preparazione della bibliografia, della composizione di indici e glossari per finire con l'arte di definire nuovi comandi compositivi per rendere più agevole il compito del compositore.

Questo testo non è un manuale; per LaTeX ce ne sono di liberi e di commerciali assai validi; a questo argomento è dedicata la prima appendice. Nello stesso tempo, dicendo che questo non è un manuale si vorrebbe sottolineare che le indicazioni qui esposte servono per affrontare la composizione con LaTeX con un approccio che guarda di più alla sostanza, alla composizione professionale, che non all'aspetto grafico del singolo dettaglio man mano che questo si presenta, come succede spesso, invece, con i vari word processor.

Per esempio in questo testo si parla di 'scatole' proprio di sfuggita, mentre in ogni manuale questo argomento richiede almeno una sezione espressamente dedicata loro. Così si parla poco o nulla di contatori o di lunghezze rigide o elastiche; sono argomenti importanti, ma che servono maggiormente per scrivere i programmi, le macroistruzioni, contenute nei file che specificano lo stile compositivo o nei file che raccolgono le macro personali. Durante la composizione non bisognerebbe mai perdersi in questi dettagli.

Piuttosto la programmazione in linguaggio TeX può diventare essenziale per l'utente che deve comporre testi dallo stile insolito o che abbia bisogno di strutture compositive particolari.

Qui, però, tranne una esposizione sommaria nel capitolo 16, non si parlerà di queste cose, ma si rinvia direttamente il lettore a manuali che trattano questo aspetto con maggiore dettaglio, (vedi l'appendice A).

L'appendice B è dedicata alle risorse per la composizione di testi filologici, mentre l'appendice C è dedicata alla simbologia e alla nomenclatura delle grandezze fisiche, nonché ai simboli codificati dalle norme ISO per l'uso in matematica; vi sono anche non pochi riferimenti al Sistema Internazionale delle Unità di Misura e alla loro 'ortografia'.

L'appendice D spiega invece in ogni dettaglio l'algoritmo che viene usato dai programmi del sistema TeX per dividere le parole in fin di riga; questo è un procedimento che in italiano funziona impeccabilmente nella quasi totalità dei casi, ma è bene conoscerne i dettagli per intervenire in quei pochi casi in cui la giustificazione non viene eseguita perfettamente oppure quando si scrive in una lingua straniera dalle regole più rigide di quelle che valgono per l'italiano.

L'appendice E, lungi dal rappresentare una semplice traduzione della corrispondente appendice del manuale di Leslie Lamport, il creatore di LaTeX, pur mantenendone l'impostazione, rappresenta, si spera, una fonte comoda e pratica per rivedere in modo ragionato ma non troppo prolisso la sintassi dei vari comandi del mark-up di LaTeX.

Capitolo 1

Composizione sincrona e asincrona

Come si è detto i word processor sono programmi per la composizione di testi dove le singole parole sono collocate nel testo composto, direttamente visibile sullo schermo; idealmente quanto si vede sullo schermo dovrebbe andare anche in stampa tale e quale, senza modifiche di sorta.

Questo genere di programmi prevede una forte interattività fra compositore e programma di presentazione o di stampa; dato il fatto che ogni tasto produce subito un effetto, così come ogni movimento e click del mouse, il tipo di composizione che viene prodotto viene detto 'composizione sincrona', proprio perché ogni azione del compositore si traduce in una immediata variazione del testo composto.

Va da sé che, perché il programma sia davvero sincrono e il ritardo fra azione e visualizzazione sia trascurabile, la forza del programma deve essere concentrata nella rapidità della presentazione. Tale caratteristica non può che andare a scapito della 'perfezione' della composizione, perché questa dipende da una elaborazione molto più accurata sul testo da comporre. È vero che oggi i programmi di videocomposizione (word processor) sono estremamente rapidi e quindi ogni anno che passa la qualità della loro composizione migliora vistosamente, tuttavia il compromesso fra velocità e qualità esiste sempre.

La 'composizione asincrona' consiste invece nell'introdurre il testo da comporre in un file senza badare al suo aspetto grafico e nel farlo fluire dentro un programma di impaginazione, dove il compositore può agire (anche in un secondo tempo) per modificare interattivamente il solo aspetto grafico. Durante questo processo può accadere di dover modificare la giustezza di alcune linee o di interi capoversi e quindi può accadere che le linee facenti parte di un capoverso o l'intero capoverso debbano venire ricomposti e giustificati diverse volte, senza limitarsi a ottimizzare la composizione della parte modificata, ma prendendo in esame l'intero capoverso; questo implica l'ottimizzazione compositiva dell'intero capoverso, non solo una aggiustatina del punto in cui si è eseguita la modifica. Ecco quindi che la composizione avviene in due tempi, l'introduzione del testo e l'ottimizzazione della composizione, avendo però a disposizione l'intero testo da trattare.

Va da sé che la composizione asincrona assicura una migliore qualità di

composizione rispetto a quella sincrona, visto che non viene tenuta in nessun conto la velocità di visualizzazione, ma la forza compositiva viene concentrata sulla qualità.

LaTeX appartiene al gruppo di programmi di composizione asincrona, ma con alcune particolarità; con un editor di testi l'autore o il compositore inserisce il testo in un file che viene successivamente elaborato con LaTeX che agisce da impaginatore; tuttavia il primo file prodotto mediante l'editor di testo non contiene solo il testo in senso stretto, ma contiene una serie di informazioni di mark-up che successivamente permettono a LaTeX di sapere che cosa sta componendo, in modo da eseguirne la composizione secondo le direttive dello stile del documento che sta elaborando.

Il testo prodotto mediante l'editor di testo è quindi una specie di programma, che contiene sia il testo da comporre, sia i comandi e le altre istruzioni necessarie per il riconoscimento della parte di testo in corso di elaborazione, sia le modalità compositive specifiche che il compositore desidera introdurre.

L'utente non deve spaventarsi con le parole 'linguaggio di programmazione'; per lo più si tratta solo di informazioni di mark-up per specificare titoli, testi speciali, composizione della matematica, e simili. I comandi e le altre specificazioni sono solitamente espresse in inglese (molto semplice, ridotto all'essenziale) evitando quasi sempre le abbreviazioni o gli acronimi.

Un breve esempio consente di capire meglio il genere di difficoltà, o meglio, di facilità di questo tipo di linguaggio di mark-up:

```
\section{Breve esempio}

Il  titolo precedente è quello di una sezione;
più propriamente la parola inglese \textit{section} in
tipografia indica ciò che in italiano viene chiamato
\textit{paragrafo}.

Questo secondo capoverso contiene una equazione numerata:
\begin{equation}
ax^2 + bx + c =0
\end{equation}
%
Esso contiene anche una tabella centrata:
\begin{center}
  \begin{tabular}{lll}
    \hline
      Nome      & relazione  & parentela \\
    \hline
      Giovanni  & è          & il papà   \\
      Ada       & è          & la mamma  \\
      Maria     & è          & la figlia \\
      Giuseppe  & è          & il figlio \\
    \hline
  \end{tabular}
\end{center}
Qui c'è la fine del capoverso che contiene
sia una formula sia una tabella.
```

L'insieme di testo e comandi contenuti nell'esempio precedente viene poi composto così:

1.1 Breve esempio

Il titolo precedente è quello di una sezione; più propriamente la parola inglese *section* in tipografia indica ciò che in italiano viene chiamato *paragrafo*.

Questo secondo capoverso contiene una equazione numerata:

$$ax^2 + bx + c = 0 \tag{1.1}$$

Esso contiene anche una tabella centrata:

Nome	relazione	parentela
Giovanni	è	il papà
Ada	è	la mamma
Maria	è	la figlia
Giuseppe	è	il figlio

Qui c'è la fine del capoverso che contiene sia una formula sia una tabella.

Nel capitoli successivi verranno spiegate tutte le informazioni di mark-up e le istruzioni usate nell'esempio. Tuttavia anche il lettore con pochi rudimenti di inglese capisce perfettamente quello che il mark-up ha specificato; le istruzioni per la composizione del materiale tabulare sono un poco più articolate, ma guardando il risultato ci si rende conto che non sono poi così misteriose.

Capitolo 2

Nozioni elementari di tipografia

Il gergo della tipografia, come tutti i gerghi professionali, contiene un certo numero di parole che hanno un significato che deriva dalla pratica o dalla tradizione.

2.1 Tipografia e dattilografia

Bisogna innanzi tutto chiarire che comporre tipograficamente è una cosa del tutto differente dalla composizione con la macchina da scrivere e perciò anche con i moderni sostituti delle macchine da scrivere, costituiti dai Word Processor.

Questi ultimi oggi sono talmente avanzati che danno l'illusione di poter eseguire una composizione tipografica; ma in realtà, a parte i compromessi fra velocità di presentazione e perfezione della composizione illustrati nel capitolo precedente, il risultato dipende moltissimo dalla professionalità del "tastierista"; costui o costei è la persona che immette il testo e che decide cosa fare, che spazi lasciare, dove mettere il materiale non testuale (per esempio le fotografie) eccetera. Se l'operatore designato con il vecchio termine di tastierista non ha abbastanza professionalità, il risultato della composizione è modesto.

La bellezza di una composizione tipografica, a parte il testo, sta nel fatto che la disposizione del materiale da leggere o da consultare non richiama su di sé l'attenzione, ma mantiene quella sobrietà che consente al lettore di recepire il messaggio senza distrazioni inutili e senza zoppicare nel leggere a causa di spaziature irregolari o continui cambiamenti di stile dei caratteri.

Vale la pena di osservare la figura 2.1 dove Herman Zapf, un grande disegnatore di caratteri contemporaneo, usa il suo font Zapfino, destinato proprio alle citazioni o alle epigrafi, per comporre una frase di Oswald Veblen che esalta la matematica (e il suo linguaggio), proprio il motivo che spinse Knuth a "inventare" la tipografia assistita da calcolatore mediante il programma TEX. Una analoga motivazione celebrativa sta alla base del biglietto di auguri del 2003 della figura 2.2 che l'associazione internazionale degli utenti di TEX ha inviato ai suoi soci.

È chiaro che questa epigrafe e questo biglietto richiamano l'attenzione su di sé; sono fatti apposta! Ma un libro completamente composto con il font Zapfino

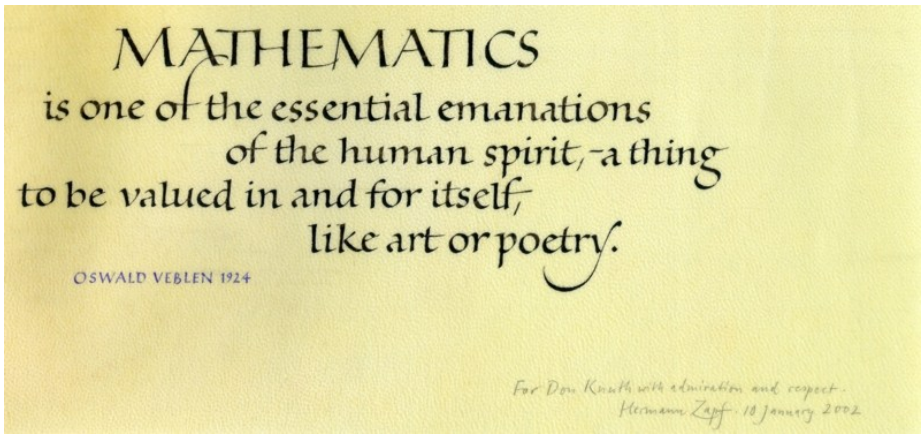

Figura 2.1: Dedica a Donald Knuth

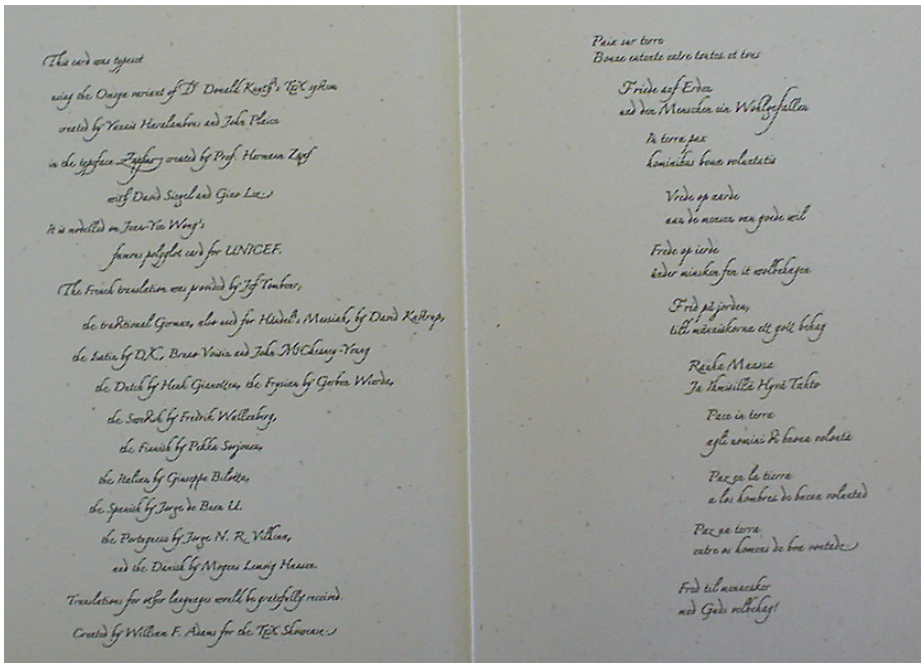

Figura 2.2: Auguri di TUG (TEX Users Group) per il 2003

sarebbe faticosissimo da leggere e il lettore sarebbe continuamente distratto dalle acrobazie calligrafiche eseguibili con questo font.

La scelta dei font, la loro grandezza, il loro stile, la loro nerezza, la scelta della distanza fra le righe, i margini, gli spazi bianchi lasciati attorno agli oggetti non testuali, sono tutte cose di competenza del "disegnatore editoriale", non di dilettanti come siamo tutti noi. LaTeX, nei limiti di quello che può fare un oggetto "inanimato" come un programma di elaborazione di dati, sostituisce sia il disegnatore editoriale sia il tipografo; non sostituisce il tastierista; se questo pensa di essere davanti ad una macchina da scrivere un po' sofisticata, nonostante tutto quello che LaTeX è capace di fare, la composizione risulterà squilibrata e disomogenea, in sostanza, l'opera di un dilettante.

La raccomandazione, perciò è la seguente: lasciamo LaTeX fare il suo mestiere! Il risultato sarà sicuramente migliore di quello che potremmo ottenere con i nostri maldestri interventi "correttivi".

2.2 Unità di misura tipografiche

In tipografia si usano molte unità di misura tipografiche che spesso, se non sempre, non hanno nulla a che fare con il sistema metrico decimale. Essenzialmente tutto viene misurato in punti tipografici o in loro multipli.

Qui nasce una prima difficoltà interpretativa. Che cosa è un punto tipografico? Ne esistono almeno tre versioni:

1. il punto tipografico anglosassone definito come la frazione 1/72,27 di un pollice, a sua volta pari a 25,4 mm. Ne segue che il punto anglosassone corrisponde a 0,3514598 mm, poco più di un terzo di millimetro. Questa è l'unità di misura usata da LaTeX, e ogni informazione metrica che LaTeX fornisce al compositore durante la sua esecuzione è sempre espressa in questo tipo di punti tipografici.

2. Il punto PostScript corrisponde a 1/72 di pollice, quindi è un poco più grande del punto anglosassone perché vale 0,35277778 mm. La differenza sembra piccola, ma quando si parla di diverse decine di punti la differenza è visibile a occhio nudo.

3. Il punto didot viene usato nell'Europa continentale; esso corrisponde a 0,3770 mm e basta una decina di punti per vedere la differenza ad occhio nudo rispetto al punto anglosassone.

La figura 2.3 presenta un confronto (relativo) delle varie scale metriche che si possono ottenere con le unità di misura descritte in questo paragrafo; se la pagina viene stampata in scala 1 : 1, allora il confronto diventa anche un confronto assoluto.

Non è il caso di preoccuparsi di queste differenze; tuttavia nasce spontanea la domanda: "Ma perché i tipografi non si sono accordati su una unità metrica?" Non so se questa risposta sia adatta: penso che la tipografia sia un'arte praticata in un ambito piuttosto ristretto dove gli interscambi erano fino alla fine dell' '800 abbastanza scarsi, quindi la possibilità di fraintendimenti era modesto. Oggi, nel mondo dell'informatica e della tipografia assistita da calcolatore, sarebbe desiderabile una maggiore uniformità. Infatti oggi si sta imponendo in tipografia il punto anglosassone e in quella assistita dal calcolatore il punto

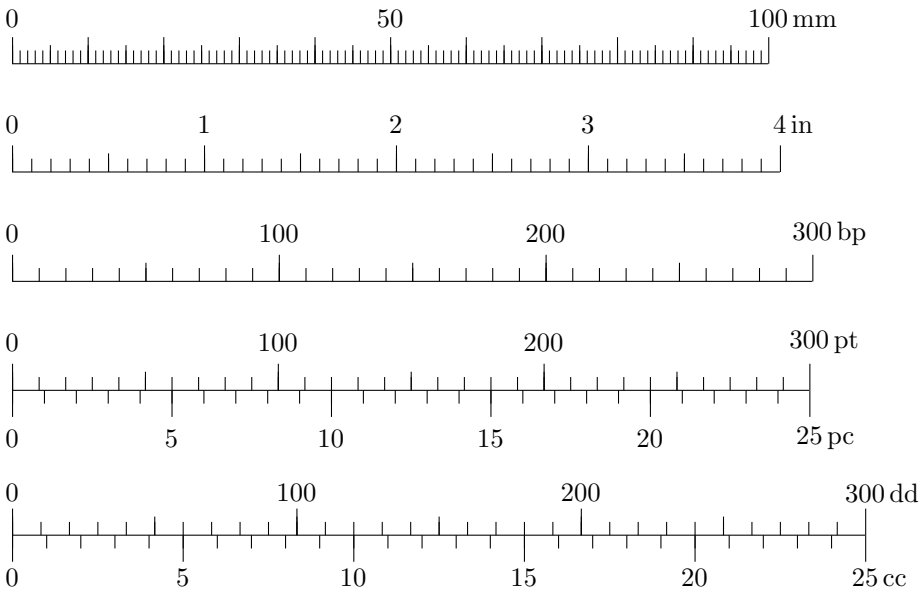

Figura 2.3: Confronto grafico fra le varie scale di lunghezza

PostScript. È un peccato che il Sistema Internazionale non sia riuscito nel compito di standardizzare una unità di misura adatta alla tipografia (per esempio 'un terzo di millimetro', ma dotato di tutti i multipli e sottomultipli decimali del caso) come invece, in parte, è riuscito a fare con il carato metrico.

Per indicare lunghezze maggiori in tipografia si usano multipli duodecimali, come il pica, che rappresenta 12 punti anglosassoni, o il cicero che rappresenta 12 punti didot. Bisogna però dire che mentre nel Nord America si usa il pica come unità di misura per indicare la giustezza (lunghezza) della riga di stampa, in Europa e in molte altre parti del mondo più o meno 'decimalizzate', Regno Unito compreso, la giustezza si indica in millimetri. Lo stesso vale per i formati delle carte da stampa; nel Regno Unito sopravvivono denominazioni verbali per i formati dei libri, ma poi la carta è misurata in millimetri. In Italia il cicero è spesso indicato col nome di 'riga', salvo poi a non essere più chiaro se si parla di una riga di testo o di una riga alta 12 punti e, ai giorni nostri, se i 12 punti siano didot o anglosassoni.

2.3 Le dimensioni dei caratteri e delle righe di stampa

La grandezza dei caratteri è indicata in punti; quando si lavorava con i caratteri mobili metallici, era chiarissimo che cosa volesse dire la parola 'corpo' dei caratteri; questi infatti erano ricavati da blocchetti parallelepipedi di metallo in cui una delle lunghezze rappresentava l'altezza del blocchetto; su una delle facce di base era ricavato in rilievo il disegno del segno da stampare. Si vedano le indicazioni costruite nella figura 2.4, dove sono evidenziate le grandezze dei caratteri mobili.

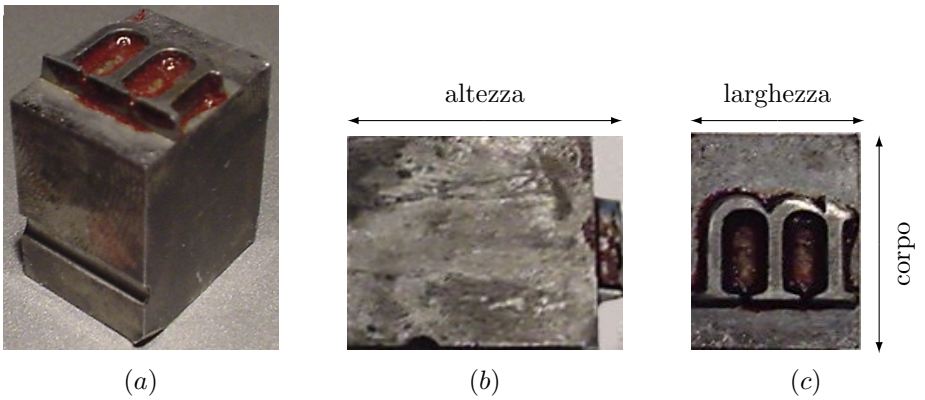

Figura 2.4: Caratteristiche dei caratteri mobili. (*a*) Visione d'insieme di un carattere mobile metallico; (*b*) vista laterale del fusto del carattere; (*c*) vista della faccia con il disegno rovesciato del segno da stampare.

La faccia di base aveva una altezza e una larghezza. La larghezza era variabile a seconda del segno che il blocchetto conteneva; è evidente che il blocchetto di una 'M' era molto più largo di quello di una 'i'. Invece l'altezza era uguale per tutti i blocchetti dello stesso alfabeto, o meglio dei blocchetti della stessa 'cassa'; la cassa era un cassetto diviso in tanti scomparti quanti erano i segni minuscoli, maiuscoli, segni di interpunzione, cifre, eccetera, che formavano la collezione dei caratteri dello stesso 'disegno'. Bene, l'altezza della base di tutti i blocchetti della stessa cassa formava il corpo e veniva misurato in punti.

Quindi un carattere di 12 pt era un segno qualunque della cassa il cui blocchetto aveva una altezza della faccia di base di 12 punti. Questo evidentemente non rappresentava l'altezza del carattere, ma, ripeto, l'altezza della faccia del blocchetto. Il fatto che i blocchetti avessero le facce di base della stessa altezza, permetteva al compositore di adagiare tutti quelli di una riga sul compositoio e poi di trasferire la riga completa sulla pagina parzialmente composta con la certezza che tutti i caratteri della riga fossero allineati correttamente.

Inoltre il fatto che tutti i blocchetti avessero la stessa altezza, come la si vede rappresentata nella figura 2.4(*b*), assicurava che essi venissero inchiostrati omogeneamente con i rulli inchiostratori e poi prendessero simultaneamente contatto con la carta da stampare e ricevessero la stessa pressione dalla pressa di stampa, così da imprimere i segni inchiostrati nella stessa maniera al fine di assicurare una nerezza omogenea dello stampato.

Oggi che i caratteri mobili metallici non si usano quasi più, che significato ha il corpo? Indica solamente l'altezza complessiva della riga di testo, spazio bianco incluso; e in pratica la distanza fra le righe di base di due righe successive quando queste non siano interlineate.

Ecco allora alcuni nuovi termini: *linea di base*, *interlinea* e *avanzamento di riga*.

La linea di base è quella linea ideale sopra la quale sembrano appoggiati tutti i segni di una riga; sotto alla riga di base sporgono le parti discendenti dei segni come quelli delle lettere g, p, q, y; sopra la linea di base sporgono tutte le lettere, ma alcune come b, d, f, eccetera, sono più alte delle altre lettere minuscole, come

le lettere a, c, e, m, n, x, eccetera. Siccome tutte queste minuscole sono alte come una lettera 'x', il loro 'occhio' (la loro altezza) in LaTeX viene indicato con il nome di *x-height*, cioè 'altezza della x'.

Sopra la riga ideale tangente alla lettera x, sporgono le lettere dotate di ascendenti e le maiuscole.

I *discendenti* sono i tratti delle lettere che sporgono sotto alla linea di base e determinano la *profondità* della riga. Gli *ascendenti* sono i tratti delle lettere minuscole che sporgono sopra la linea tangente alla 'x'. Il più alto ascendente è spesso quello dalla lettera 'h'; talvolta sporge sopra all'altezza delle maiuscole, talvolta è alto quanto le maiuscole. In ogni caso in ogni riga l'altezza della riga è determinata dall'ascendente più alto. Non necessariamente la somma della profondità e dell'altezza di una riga di testo è uguale al corpo.

Come già detto, di fatto il corpo è la distanza di due righe di base composte senza inserire alcuna interlinea. Poiché non necessariamente una data linea contiene tutti i caratteri con i discendenti e tutti quelli con gli ascendenti, il corpo può sembrare maggiore dell'altezza complessiva di una riga, ma in generale, anche se i segni di due righe successive composte senza interlinea non interferiscono fra di loro, la mancanza di interlinea spesso rende il blocco di testo troppo compatto e faticoso da leggere.

Per questo motivo si inserisce una interlinea di un paio di punti, o proporzionalmente di più per i corpi più grandi. Tradizionalmente l'interlinea era quella striscia di metallo che veniva inserita fra due righe successive per distanziarle un poco; in inglese sia chiama *leading*, da pronunciare come 'lead' (piombo), perché queste strisce erano composte di piombo (o della stessa lega di piombo usata per i caratteri da stampa).

Di conseguenza l'*avanzamento di riga* rappresenta la distanza effettiva fra le linee di base di due righe successive composte interponendo l'interlinea. Si dirà quindi che un certo testo è composto in corpo 12/14 per dire che i caratteri hanno un corpo di 12 punti e che l'avanzamento di riga è di 14 punti, il che implica la presenza di una interlinea di 2 punti. A questo proposito si veda la situazione appena descritta rappresentata nella figura 11.2.

2.4 Le particolarità dei caratteri

Ognuno avrà notato che i caratteri da stampa si dividono sostanzialmente in due grosse categorie, ma in realtà essi sono distinti da una varietà di caratteristiche che permettono moltissime diverse classificazioni.

La differenza fra i caratteri usati dalle vecchie macchine da scrivere, a spaziatura fissa, e i caratteri da stampa vera e propria, a spaziatura variabile, credo che sia evidente a tutti. Oggi i caratteri a spaziatura fissa sono quasi del tutto scomparsi, ma trovano importanti applicazioni, specialmente di carattere informatico, ma non solo, quando è necessario evidenziare con il tipo di segno usato certi scritti tecnici, come i brani di linguaggio di programmazione, che non devono essere giustificati e non devono avere nomi o parole chiave divisi in sillabe in fin di riga, ma la cui spaziatura fissa aiuta il lettore a distinguere le variabili dagli operatori, quindi a leggere il codice con più facilità.

I caratteri a spaziatura variabile sono più comodi da usare nei testi giustificati e consentono in generale una lettura più agevole, specialmente se sono caratteri con grazie. Le grazie sono quei piccoli segni alle estremità delle aste,

solitamente con andamento orizzontale, che rendono maggiormente definito il contorno dei singoli segni e, specialmente le grazie inferiori, consentono di allineare meglio la direzione della lettura lungo la riga di testo. Quindi una seconda classificazione riguarda la presenza di grazie (*serif* in inglese) o la loro assenza (*sans serif*).

I caratteri non hanno poi la stessa forma; ognuno è in grado di distinguere la forma dei caratteri tondi (*roman* in inglese) dai caratteri *inclinati* (*slanted* in inglese); questi a loro volta si distinguono dal fatto che sono ottenuti dal tondo semplicemente inclinando le aste, oppure sono disegnati apposta con andamento più corsivo (*italic* in inglese), sebbene di questo corsivo sia possibile disporre di una versione non inclinata.

Inoltre un alfabeto può contenere sia le lettere maiuscole, sia quelle minuscole dalla forma completamente distinta da quella delle maiuscole, oppure può contenere un alfabeto minuscolo in cui le lettere minuscole assomigliano ad una versione ridotta delle maiuscole; in quest'ultimo caso si è in presenza del maiuscoletto (*small caps* in inglese).

La nerezza dei segni può assumere livelli diversi; le collezioni di caratteri più ricche hanno diverse gradazioni di nero: chiarissimo, chiaro, normale, neretto, nero, nerissimo. I font per tipografia assistita dal calcolatore spessissimo hanno solo una versione normale e una nera, ma le collezioni più ricche possono avere diverse gradazioni.

Per altro gli stessi segni, con il medesimo grado di nero, possono avere dei disegni ristretti (*condensed*) o allargati (*extended*) rispetto alla dimensione normale.

Una caratteristica che invece non è visibile è la codifica; questo è un concetto tipicamente informatico, nel senso che ogni segno si trova descritto dentro un file insieme agli altri segni dello stesso alfabeto, e per recuperare quel segno dal file è necessario conoscere il suo 'indirizzo'. Esistono moltissimi modi di indirizzare i caratteri (i singoli segni) all'interno di un file e la cosa dipende anche dal numero di segni che il file contiene. Oggi la codifica UNICODE permette di avere file enormi che contengono un numero enorme di segni, disegnati in modo più o meno uniforme anche se appartengono ad alfabeti diversi. Non è raro il caso di file relativi ad una sola collezione che contengono l'alfabeto latino con i suoi segni numerici e di interpunzione, l'alfabeto greco; l'alfabeto cirillico, l'alfabeto ebraico, l'alfabeto arabo, una certa collezione di segni cinesi, giapponesi, coreani, raccolte di segni speciali, come i simboli matematici o i segni astrologici, e chi più ne ha più ne metta, e non è difficile arrivare a qualche decina di migliaia di segni.

La codifica è importantissima, perché si scrive assumendo una certa codifica e si vede sullo schermo il testo correttamente rappresentato; ma poi se si cambia font, si rischia di non avere più la possibilità di decifrare il messaggio; a me è capitato più di una volta di aprire con Word dei documenti composti con un'altra versione di Word e trovare che i segni originali erano stati sostituiti dai segni matematici; la lettura evidentemente era diventata impossibile.

La cosa più importante per i professionisti che si occupano della grafica editoriale consiste nella scelta dei font; due font di forma tonda diritta, nerezza media, estensione normale, con lo stesso corpo e relativi allo stesso alfabeto possono apparire completamente diversi a seconda del rapporto fra lo spessore delle aste spesse e quelle sottili, a seconda della forma delle grazie, a seconda dell'inclinazione dell'asse ottico (non l'inclinazione delle aste), può apparire

completamente diverso e può dare luogo ad una lettura più facile oppure più faticosa, se composta in righe con lo stesso avanzamento e della stessa lunghezza.

Questo testo è composto con i caratteri della collezione Computer Modern, con codifica a 128 caratteri del sistema TEX[1]; per lo più viene usata la forma di nerezza media e larghezza normale; il corpo normalmente usato è il corpo 10 composto con un avanzamento di riga di 12 punti. Talvolta si usa il corsivo per mettere in evidenza delle parole o per scrivere quelle in inglese. I titolini dei paragrafi sono scritti in tondo diritto, nerissimo esteso in corpo 14,4 con un avanzamento di 20 punti; l'avanzamento dei titolini in realtà non è molto importante perché quasi tutti si svolgono su una sola riga.

2.5 I contrografismi

Logicamente la pagina stampata contiene i segni che convogliano il messaggio, che vengono collettivamente denominati *grafismi*. I *contrografismi* sono le parti della pagina prive di segni, eventualmente, solo colorati, che svolgono una funzione essenziale nel disegno grafico della pagina, ma specialmente nella qualità della composizione, la cui lettura viene agevolata dal loro uso giudizioso.

Non solo l'interlinea è un contrografismo, ma anche la rientranza dei capoversi, gli spazi sopra e sotto alle figure, alle formule, ai titolini; i quattro margini di ciascuna pagina, quello superiore, quello esterno, quello inferiore e quello interno.

Questi ultimi, infatti, determinano la posizione della gabbia che contiene il testo e talvolta anche la *testatina* e/o il *piedino* (l'intestazione della pagina e la riga di piè di pagina; *header* e *footer* in inglese); se questi due elementi sono vuoti o quasi vuoti essi contribuiscono più alla grandezza dei margini che alla grandezza della gabbia.

I margini, specialmente quelli esterni, secondo un disegno grafico abbastanza tradizionale, quando sono sufficientemente ampi possono accogliere le note marginali; quindi le note possono trovare posto sia in calce alla pagina, sia nei margini; in qualche scritto di stile critico–letterario esse possono essere anche collocate in fondo a ogni capitolo.

Ma i margini possono accogliere anche parte delle figure o le didascalie delle figure; non si tratta quindi di spazio 'sprecato' ma di spazio che oltre a dare un tono alla pagina può svolgere funzioni ausiliarie al testo in modo creativo e spesso gradevole.

2.6 Le parti di alcuni documenti a stampa

Un libro, generalmente è la forma di stampato più complessa e a questa mi riferirò nel descrivere le parti.

Un rapporto è uno stampato che può assumere la forma di un libretto, ma generalmente ha meno pretese stilistiche, sia per la relativa brevità sia per il

[1] Inizialmente si era usata la collezione European Computer Modern con codifica a 256 caratteri, ma poi si è ripiegato sulla "vecchia" collezione con codifica a 128 caratteri, perché alcune delle persone che hanno collaborato alla stesura si sono trovate con una installazione del sistema TEX ... troppo personalizzata, al punto da non essere più compatibili. Sono cose che succedono quando si usa uno strumento troppo potente.

contenuto spesso assai tecnico (il che non vuol dire solo ingegneristico, ma anche giuridico, economico, eccetera).

L'articolo è invece un tipo di stampato, di solito assai breve e molto conciso; la concisione implica che non viene 'sprecata' un'intera pagina per presentare il titolo; spesso è scritto su due colonne, il che è un artificio per usare gabbie di testo più larghe, capaci di contenere righe che, se scritte a piena pagina, risulterebbero troppo faticose da leggere.

Un libro può essere confezionato *incassato* (con copertina rigida) o *brossurato* (con copertina morbida incollata); all'interno della copertina esso comincia con un foglio generalmente privo di ogni scritto, seguito da una pagina, sulla cui facciata di destra (il *recto*) può essere riportato l'*occhiello*; il retro (il *verso*) è sempre senza testo; la pagina successiva sul recto presenta il titolo, mentre sul verso presenta le informazioni di carattere legale richieste, appunto, dalla legge.

La successiva pagina può contenere una dedica sul recto, mentre sul verso è senza testo; se la dedica manca, manca l'intero foglio.

Segue poi sul recto del foglio successivo una sezione non numerata generalmente intitolata 'Presentazione' o con un titolo equivalente. Spesso non supera una pagina, ma può anche occupare qualche pagina.

La successiva sezione, che si apre sul recto, contiene l'indice generale a cui possono seguire, aprendosi sul recto delle pagine, l'elenco delle figure, l'elenco delle tavole, l'elenco delle illustrazioni fuori testo; ovviamente la presenza di questi elenchi dipende dal tipo di libro, dal suo contenuto. Nei libri di carattere letterario, come i romanzi, spesso l'indice si trova in fondo al libro e in Italia era questa la consuetudine di ogni tipo di libro molte decine di anni fa. Oggi molti romanzi e tutti i libri con indici molto strutturati presentano l'indice nelle prime pagine, come descritto sopra.

L'Introduzione segue queste parti iniziali, ma in Italia il materiale iniziale, *front matter* in inglese, termina solitamente prima dell'Introduzione. Le pagine che contengono il materiale iniziale sono generalmente numerate con numeri romani; terminato il materiale iniziale, la numerazione delle pagine ricomincia da 1 e viene scritta con cifre arabe fino alla fine del libro[2].

Dopo il materiale iniziale comincia il corpo del testo, *main matter* in inglese.

Il corpo del libro è spesso diviso in parti, ma è sempre diviso in capitoli; ogni capitolo è spesso diviso in sezioni gerarchiche che in italiano si chiamano paragrafo, sotto-paragrafo, sotto-sotto-paragrafo, capoverso e sotto-capoverso; in inglese esse si chiamano rispettivamente *section, subsection, subsubsection, paragraph, subparagraph.* Si noti la presenza dei falsi amici 'paragrafo' e 'paragraph'!

Il capoverso è quel tratto di testo che comincia (generalmente con una rientranza) con una lettera maiuscola e termina con un 'punto e a capo'. Un capoverso può contenere diversi periodi, ciascuno diviso in diverse frasi, a loro volta... ma qui si sconfinerebbe nella grammatica della lingua. Spesso singoli periodi, o anche solo delle frasi di una certa lunghezza, sono numerati e, come sotto-capoversi, a seconda dello scritto possono chiamarsi versetti, oppure commi. In generale i versetti sono solo numerati, mentre i commi possono avere anche un breve titolo; in questo caso essi cominciano su una nuova riga di

[2] LaTeX di default usa i numeri romani minuscoli scrivendo per esempio l'equivalente di 1666 come 'mdclxvi'; in Europa, tranne nelle isole britanniche, generalmente si preferisce il maiuscoletto; questa modifica alla classe `book` è una delle pochissime che sono state eseguite per la composizione di questo testo.

testo, ma il paragrafo (che allora sarà un 'articolo' di carattere legale) viene distinto dal fatto che viene inserito un visibile contrografismo fra un articolo e il successivo.

Finito il corpo del testo, cioè dopo l'ultimo capitolo, comincia il materiale finale, la *back matter*, la cui numerazione araba prosegue quella del corpo del testo, ma la numerazione dei capitoli ricomincia da 1, o meglio da A, visto che si usa una numerazione diversa, per esempio letterale; ma mentre i capitoli e le parti cominciano sempre sul recto delle pagine, nel materiale finale i capitoli possono cominciare anche sul verso.

Il materiale finale contiene la Bibliografia, gli eventuali glossari, le appendici (capitoli numerati con lettere), e termina, se c'è, con uno o più indici analitici, i quali vengono evidentemente composti per ultimi per evitare che le loro informazioni siano errate a causa delle correzioni eventualmente apportate nel corpo del testo.

L'ultimissima pagina, sul verso, potrebbe contenere il colophon, una pagina, cioè, nella quale sono descritte le particolarità compositive del testo, i materiali usati e simili altre informazioni di carattere tipografico. Vengono poi lasciate un paio di pagine bianche per poter inserire i risguardi posteriori a cui è incollata la parte posteriore della copertina.

Per ovvi motivi economici le varie pagine di un libro sono ottenute piegando un certo numero di volte dei grandi fogli; queste *segnature* contengono evidentemente un numero di facciate pari ad una potenza di 2; ci saranno perciò libri in ottavi, in sedicesimi, in trentaduesimi, raramente con un numero di facciate superiore, perché lo spessore delle segnature produrrebbe problemi.[3] Resta il fatto che, tolti i risguardi, le pagine interne devono essere un multiplo del numero di pagine di una segnatura; al massimo l'ultima segnatura di un libro formato da segnature di 32 pagine, potrà essere una segnatura da 16 pagine o anche di 8 pagine, ma questo aumenta leggermente i costi di produzione. Nel predisporre la produzione di un libro le tipografie generalmente sono più attente al numero di pagine che non al loro contenuto (il contenuto è responsabilità dell'autore e della casa editrice); ma è bene che anche l'autore conosca questi problemi in modo da sapersi regolare se deve produrre un testo pronto per la stampa, cioè un file PDF da trasmettere alla tipografia che non vi mette mano se non per ottenere dal file le lastre per la stampa delle due facce dei grandi fogli che formeranno ciascuna segnatura.

2.7 Osservazioni finali

È chiaro che queste poche pagine non sono in grado di dare altro che una breve panoramica della terminologia tipografica, ma spero che il lettore, con queste poche conoscenze, sia stimolato ad osservare con più attenzione i libri che ha per mano e cerchi di valutare i vari elementi descritti, sia in relazione ai caratteri, sia in relazione agli spazi e alla suddivisione dello stampato.

I nomi inglesi che ho indicato servono anche per conoscere il significato di molti comandi o istruzioni del linguaggio LaTeX, che usa quelle parole o quelle brevi locuzioni, ridotte ad una sola stringa senza spazi, per indicare esattamente quegli oggetti individuati dai nomi inglesi.

[3] Con particolari piegature si potrebbero avere anche segnature il cui numero di facciate corrisponde ad una potenza di due moltiplicata per tre; per esempio si potrebbero avere segnature di 6 pagine, di 12 pagine, eccetera; questi tipi di piegature non sono molto frequenti nei libri, ma si trovano usate in fascicoletti propagandistici e altri simili opuscoli.

Capitolo 3

Installare il sistema TEX

L'installazione del sistema TEX può essere molto semplice o molto complicata; dipende da quali strumenti si usano e da come l'utente sa usare efficacemente il sistema operativo del proprio PC o laptop.

Si distinguono tre casi:

1. Sistemi operativi Windows da Windows98 in poi con processore a 32 bit;
2. Sistemi operativi Linux;
3. Sistema operativo Mac OS X di qualunque sotto-versione.

Qui si parlerà delle versioni freeware; per le versioni commerciali l'utente deve valutare bene il rapporto prestazioni/prezzo in relazione alle proprie esigenze. Probabilmente è preferibile cominciare con una versione gratuita, per poi passare ad una versione commerciale se e quando si sarà constatata le reale necessità di quelle caratteristiche in più che la versione commerciale offre rispetto alle versioni gratuite.

Va notato che l'organizzazione degli utenti di TEX, TEX Users Group (TUG), offre la possibilità di scaricare dal suo sito `www.tug.org` l'immagine del disco TEXlive, che, comunque, viene inviato gratuitamente a tutti i suoi soci; questo disco, dati i tempi per il download, può essere anche ordinato alla sede di TUG e lo si può ottenere praticamente al costo della duplicazione del disco o poco più. Questo disco contiene le distribuzioni per tutti i principali sistemi operativi e per tutte le macchine più diffuse; quello che si esporrà qui di seguito può essere ritrovato in quel disco.

3.1 Installazione su Windows98 e successivi

La forma più semplice per installare il sistema TEX su una macchina Windows che usi un processore a 32 bit è quella di procurasi il CD proTEXt; detto CD può anche essere scaricato come un grande file compresso da uno dei siti dove si trova tutto il materiale relativo al sistema TEX. Per esempio ci si può riferire al sito

`http://www.ctan.org/tex-archive/systems/win32/protext/`

e ai file che vi sono contenuti; si tratta di due piccoli file di poche decine di byte e di un grande file di più di 400 MB, quindi è necessario disporre di una connessione molto veloce. Esiste anche una immagine da masterizzare su un CD, che può essere usata direttamente dal CD senza installare nulla, ma che consente di provare l'intero sistema.

Estratta l'immagine dal file auto-estraente scaricato dal sito, bisogna cercare il file PDF `protext-install.pdf` e seguire religiosamente le istruzioni lì contenute; in realtà basta cliccare gli opportuni bottoni che attivano tutte le operazioni necessarie alle installazioni di tutte le varie parti del pacchetto. È bene sottolineare che insieme al sistema TEX il processo di installazione provvede a installare o a re-installare almeno uno shell editor (anche se non necessario, è però molto utile per gestire i file sorgente con cui interagire con LATEX) e per installare o re-installare programmi accessori come ghostscript, ghostview, eccetera.

Terminata questa installazione guidata, il sistema è pronto per l'uso.

3.2 Installazione su Linux

Spesso il sistema TEX risulta già installato insieme al sistema operativo; in ogni caso esso è presente nei dischi di installazione del sistema operativo e dei suoi applicativi, dai quali può venire installato come qualunque altro programma di Linux, avvalendosi eventualmente di interfacce grafiche come per esempio Kpackage tipico delle versioni Linux che usano il KDE desktop. Generalmente non c'è altro da fare, salvo eventualmente lanciare il programmino texconfig per eseguire una configurazione locale, per esempio, per preconfigurare il sistema per l'uso della lingua italiana; per esperienza so che la distribuzione Debian non ha bisogno di questo passo perché il sistema esce dal processo di installazione già predisposto per trattare tutte le lingue di cui il sistema TEX è capace; dall'edizione 2007 del disco di TEXlive, anche questa distribuzione nasce configurata per gestire tutte le lingue che LATEX è capace di gestire. In ogni caso il disco TEXlive sembra fatto apposta per l'installazione su macchine Linux, sebbene contenga anche le distribuzioni valide per gli altri sistemi operativi.

Il pacchetto di installazione non installa lo shell editor necessario per operare sui file sorgente da trattare con LATEX; si può usare l'onnipresente e onnipotente programma emacs con il plug-in `auctex`; se lo si trova troppo complicato, si può installare Kile ottimo e facile da usare. I programmi accessori, come ghostscript, fanno già parte di serie del sistema Linux. Il disco TEXlive contiene anche diversi shell editor fra cui scegliere quello che sembra più adatto alle proprie esigenze.

3.3 Installazione su macchine con sistema operativo Mac OS X

Da quando esiste il sistema operativo Mac OS X e la macchina a sua volta usa un processore Intel Dual Core oppure PowerPC, esistono due versioni di MacTEX, una per ciascuna delle macchine dotate di quei processori. Bisogna quindi stare attenti a scaricare la versione giusta per il proprio processore.

Dal sito

`http://www.ctan.org/tex-archive/systems/mac/mactex/`

si può scaricare il pacchetto autoinstallante MacTEX che si installa praticamente da solo e risulta già preconfigurato per l'italiano. Da marzo 2007 la distribuzione MacTEX si affida a TEXlive, per cui ne condivide tutte le proprietà; in particolare la prima installazione è predisposta per lavorare con tutte le lingue che LATEX è capace di gestire.

Il pacchetto installa anche lo shell editor TEXShop necessario per lavorare sui file sorgente necessari per usare LATEX; il sistema operativo è già dotato dei programmi accessori per visualizzare il risultato della composizione e per procedere alla stampa.

3.4 I programmi accessori

La composizione differita consente di usare un programma specializzato per ciascuna operazione di composizione, in modo da usare sempre il prodotto migliore per ciascuna fase di 'lavorazione' del documento.

I programmi accessori più importanti sono i seguenti:

1. Lo shell editor serve per generare e/o modificare i file sorgente da dare in pasto a LATEX.

2. Siccome il file di uscita 'standard' di LATEX ha il formato DVI (DeVice Independent) è necessario disporre del programma specializzato per rendere il frutto della composizione visibile e leggibile sullo schermo, o per renderlo stampabile con una varietà di dispositivi diversi, che possono andare dalle stampanti alle fotocompositrici. Ogni sistema TEX è corredato da un programma per visualizzare il file DVI, spesso utile anche per stampare. Talvolta è necessario convertire il file dal formato DVI al formato PostScript, e per questo formato funziona egregiamente il programma ghostscript; ma per visualizzare il file così prodotto è utile disporre del programma ghostview, meglio conosciuto come gview o anche come gv; questo programma è una comoda interfaccia grafica per maneggiare un file PostScript al fine di visualizzarne il contenuto logico, per stamparlo e anche per trasformarlo in formato PDF. Chi esegue il lavoro vero è sempre Ghostscript, ma gview rende molto più agevoli tutte le operazioni, perché Ghostscript è un poderoso programma che però richiede di essere azionato dalla linea di comando (dalla finestra comandi di Windows o dalla finestra Terminal, Xterm, o Console dei sistemi UNIX, Linux e Mac OS X).

3. I file in formato PDF sono maneggiabili mediante gli appositi programmi caratteristici di ogni sistema operativo; per tutti la Adobe produce il programma freeware Adobe Reader, che consente di visualizzare i file nel formato PDF e di ottenerne alcune informazioni. Se si desidera la piena funzionalità sarebbe necessario acquistare da Adobe il programma Adobe Acrobat (eventualmente nella versione professional), oppure altri programmi simili; Adobe Acrobat consente di eseguire qualche ritocco al contenuto del file, di estrarne delle intere pagine o delle figure, di eseguire il ritaglio di parti della pagina, ovvero di scontornare le immagini.

4. Sono utili anche diversi altri programmi di conversione di formato, non necessariamente già in dotazione del sistema operativo in uso, ma che tornano utili in diverse circostanze; per esempio jpegtops trasforma le immagini

dal formato JPG (o JPEG) al formato PostScript, o meglio, Encapsulated PostScript. Il programma ps2pdf trasforma i file dal formato encapsulated PostScript al formato PDF. Il programma gimp, nato per le piattaforme UNIX/Linux e ora disponibile anche per le altre piattaforme Windows e Mac, serve per modificare le immagini di quasi tutti i possibili formati e di salvarle in qualunque degli innumerevoli formati che è capace di gestire.

Questi programmi accessori sono utili di per sé e quindi possono essere usati anche indipendentemente dal sistema TEX, tuttavia risultano molto utili per svolgere alla perfezione certe funzioni che solo un programma specializzato sa fare, certamente meglio di quanto potrebbe fare un programma generico dalle 'troppe' funzioni.

Tutti i programmi menzionati, tranne Adobe Acrobat, sono disponibili come freeware e sono scaricabili dalla rete adatti per essere usati sulle specifiche piattaforme di lavoro a disposizione.

3.5 LATEX e pdfLATEX

È bene che il lettore sappia che quando si dice LATEX si intende non solo il programma LATEX vero e proprio, ma anche il programma pdfLATEX. Quest'ultimo ha la caratteristica che il suo formato di uscita è direttamente il formato PDF e non sono necessarie conversioni di nessun genere come è stato più volte menzionato nel paragrafo precedente.

I file sorgente elaborabili con LATEX o con pdfLATEX sono in generale identici, o possono essere resi tali. Per cui quando si parla di un file elaborabile con LATEX, si intende che è stato scritto facendo uso del metodo di mark-up tipico di LATEX, e che il file è elaborabile indifferentemente anche da pdfLATEX. Sarebbe desiderabile che i file di questo genere avessero l'estensione `.ltx`, invece che l'estensione `.tex`, ma per vari motivi questa convenzione non viene quasi mai rispettata.

A causa dell'uso di pacchetti di estensione specializzati, un file apparentemente conforme alla grammatica di mark-up di LATEX può risultare incompatibile per essere elaborato con pdfLATEX. Ciò succede raramente, ma non è impossibile che accada; sarebbe quindi desiderabile evitare tali pacchetti di estensione, ricorrendo a pacchetti meno specializzati ma adeguati per la particolare funzione per la quale essi vengono invocati. Questa soluzione può essere usata nella stragrande maggioranza dei casi.

Talvolta l'incompatibilità fra LATEX e pdfLATEX non dipende dal file sorgente, ma dai file inclusi; ciò può accadere facilmente con i file che contengono delle immagini; è per questo che sono molto utili quei programmi accessori che consentono di trasformare una immagine da un formato all'altro; a questo proposito si faccia riferimento al capitolo 8.

3.6 Altri programmi di composizione del sistema TEX

Vale la pena ricordare in questo capitolo la presenza di altri 'fratelli' di LATEX e pdfLATEX. Essi hanno campi di applicazione in parte complementari a quelli di

LaTeX e spesso possono venire installati di default quando si installa il sistema TeX.

3.6.1 Plain TeX

Il TeX semplice, 'plain' in inglese, è il programma formato dall'insieme di comandi predisposto dall'autore Knuth e con il quale, tra l'altro, egli continua a comporre tutti i suoi libri.

Come linguaggio di mark-up è molto meno strutturato di LaTeX, tanto che talvolta il suo insieme di comandi viene qualificato come 'prescrittivo', invece che 'descrittivo'; LaTeX ha un insieme di comandi di mark-up decisamente più descrittivo, ma non è completamente descrittivo; così plain TeX ha un insieme di comandi di gran lunga molto prescrittivo, e solo qualche comando di tipo descrittivo.

Plain TeX oggi viene usato poco, ma in qualche raro caso si incontrano dei file sorgente scritti da altri che possono essere composti ricorrendo a plain TeX. In questo caso gli shell editor sono in grado di eseguire questo compito con la semplice pressione di uno dei vari pulsanti presenti nei vari menù, quindi l'utente 'normale' non ha bisogno di conoscere plain TeX. Sappia tuttavia che l'insieme di comandi di LaTeX costituisce una enorme sovrastruttura sui comandi di plain TeX e che in generale i comandi di plain TeX sono usabili sotto LaTeX e generalmente producono lo stesso effetto.

Per la documentazione ci si può rivolgere direttamente alla pietra miliare costituita dal libro e manuale di Knuth, che non dovrebbe mancare dalla scrivania di qualunque utente avanzato di LaTeX.

3.6.2 Il programma ConTeXt

Il programma ConTeXt assomiglia molto vagamente sia a plain TeX, sia a LaTeX. Esso è un programma molto strutturato, ma tutto sommato semplice come plain TeX; la differenza sta nel fatto che virtualmente ogni comando accetta che vengano specificate delle opzioni per cui la personalizzazione (customizzazione) della composizione avviene in modo più semplice, molto più semplice rispetto a quanto sarebbe necessario fare con plain TeX.

Esso permette di descrivere il testo da comporre in modo molto più strutturato anche rispetto a LaTeX, e i libri e gli altri documenti composti con questo programma sono generalmente molto ben riconoscibili per l'altissima qualità del risultato finale. Tutto ciò avviene grazie alle opzioni previste e attivate dai vari comandi usati per la descrizione del testo. Gli autori hanno spesso affermato che essi preferiscono la libertà di definire qualunque dettaglio compositivo ma non volevano perdere le funzionalità offerte da LaTeX; direi che ci sono riusciti magnificamente e i risultati si vedono. Tuttavia il linguaggio è così differente da quello di LaTeX che la maggior parte degli utenti continua a usare LaTeX.

ConTeXt è stato arricchito da molti altri pacchetti di estensione in modo che può svolgere compiti piuttosto specializzati in modo molto comodo.

Si sta sviluppando un programma, chiamato Aleph, il cui compito è quello di estendere ConTeXt così da poter gestire i caratteri UNICODE; questo fatto metterebbe ConTeXt e Aleph in concorrenza stretta con Omega e Lambda (vedi sotto), altri due programmi di solito installati di default, che si comportano rispettivamente come plain TeX e come LaTeX, ma gestiscono font con centinaia

o migliaia di caratteri, quando invece TEX, ConTEXt e (pdf)LATEX si limitano a raccolte di font di non più di 256 caratteri ciascuna.

3.6.3 I programmi estesi

I programmi eTEX, eLATEX, pdfeTEX e pdfeLATEX; sono altre estensioni di default di TEX e LATEX che dal 2005 sostituiscono i corrispondenti programmi senza la 'e', arricchendo i rispettivi progenitori di ulteriori possibilità; anzi, quando uno crede di usare LATEX in realtà sta usando eLATEX, anche se non sfrutta le nuove possibilità offerte. Insomma i programmi con la 'e' sono *downward compatible* con i rispettivi progenitori. Tutti i pacchetti di estensione che fino ad oggi sono stati sviluppati per TEX, LATEX e i loro 'fratelli' sono quindi usabili anche con i programmi con la 'e'; per altro non ho ancora visto sviluppato nessun pacchetto di estensione che richieda espressamente l'uso dei programmi estesi e non sia compatibile con i progenitori.

C'è da sperare, però, che tutti i pacchetti di estensione esistenti vengano via via aggiornati, non tanto per dotarli di nuove funzionalità, ma per sfruttare le estensioni dei 'modi' di esecuzione che possono svolgere nativamente certe funzioni, che fino ad oggi non sono sfruttate oppure che sono simulate con sequenze di istruzioni TEX o LATEX.

Vale la pena di sottolineare un fatto che a prima vista potrebbe apparire come superfluo, ma non lo è; è vero che fino ad oggi i programmi generalmente distribuiti su CTAN non sfruttano le nuove potenzialità o funzionalità de programmi con la 'e', anche se questi sono i motori veri di ogni variante dei programmi di composizione del sistema TEX. Tuttavia questi programmi cominciano ad apparire; il primo che lo scrivente ha incontrato è proprio formato dalla classe `artexnica` predisposto per comporre gli articoli della rivista ArsTEXnica dell'associazione GUIT. Questa classe di composizione richiede la modalità estesa dei programmi di composizione; di default, invece, l'ultimo aggiornamento di MacTEX2007 che lo scrivente ha installato predisponeva il sistema per usare, sì, i motori di composizione eTEX e pdfeTEX, senza però attivare la modalità estesa. La situazione si è risolta ricostruendo tutti i file di formato in modalità estesa dando dalla finestra comandi, o console, o terminal, o xterm e con i privilegi dell'amministratore il comando:

```
fmtutil -all
```

che ha, appunto, ricreato indistintamente tutti i file di formato in modalità estesa. Forse pochi lettori di questo testo si troveranno nella necessità di fare altrettanto, ma è bene sapere che, all'occorrenza, questa breve e semplice azione risolve tutti i problemi.

3.6.4 I programmi Omega e Lambda, Aleph e Lamed

Il programma Omega, continuamente in sviluppo, cerca di sostituire il programma TEX in modo da togliergli tutte le limitazioni che ha, per esempio quella di trattare font con non più di 256 segni, e di arricchirlo di altre funzionalità. Tra le altre funzioni consente anche la composizione con alfabeti retrogradi; con l'arabo, che è sostanzialmente ancora una scrittura calligrafica molto legata, consente di eseguire legature molto complesse che coinvolgono anche quattro o cinque segni.

Il programma Lambda è, per così dire, la versione LaTeX di Omega; ne accetta tutte le estensioni e mette in grado il compositore di riferirsi ad una sintassi simile a quella di LaTeX.

Il programma Aleph è una versione di ConTeXt che sfrutta il motore Omega; si ottengono così i vantaggi dell'uno e dell'altro programma; l'autore di Aleph sta lavorando anche al programma Lamed, versione di Aleph LaTeX-compatibile.

Questi programmi sono in via di sviluppo; nonostante gli eccellenti risultati, non sembrano ancora avere raggiunto il favore della gran parte dei compositori e vengono usati solamente in ambienti ristretti; ovviamente esistono le eccezioni che confermano la regola!

3.6.5 Il programma XeTeX

Il programma XeTeX, ora disponibile per tutte le piattaforme di elaborazione e con qualunque sistema operativo, è una ulteriore estensione che riunisce molte delle funzionalità dei programmi precedentemente descritti, ma in più consente di usare tutti i font presenti sul particolare calcolatore dove venisse impiegato, purché questi siano font di tipo PostScript, oppure TrueType oppure OpenType. Non occorrono estensioni particolari per l'uso di questi font, perché il programma sfrutta i comandi del sistema operativo per gestire i font disponibili e quindi una delle operazioni relativamente meno semplici da fare con LaTeX per gestire i font viene superata in un solo colpo. Il programma non è ancora diffusissimo, ma si sappia fin d'ora che gestisce tutti i possibili font di qualunque alfabeto, anche i sistemi di ideogrammi e gli alfabeti retrogradi come quello ebraico o quello arabo. Mescolare lingue diverse scritte con sistemi di caratteri completamente diversi diventa quindi una operazione semplicissima.

Ovviamente esiste anche la versione LaTeX-like di XeTeX, e non è particolarmente difficile usarla pur di riferirsi al particolare manuale che l'accompagna, se non altro per prendere nota delle differenze rispetto al sistema LaTeX standard.

Capitolo 4

LaTeX: prime nozioni

4.1 Introduzione

Verranno ora esposte le prime nozioni per usare LaTeX e per comporre semplici documenti di solo testo. Sarà necessario mostrare esempi di codice sorgente, così come sarà necessario descrivere la 'sintassi' di alcuni comandi.

Il codice e i vari comandi saranno composti con il font della famiglia a spaziatura fissa, per esempio si scriverà `\documentclass`. La grammatica prevede che ogni comando possa avere degli argomenti il cui significato logico viene indicato da un nome o da una semplice locuzione scritta in corsivo e racchiusa fra parentesi 'acute', per esempio ⟨*classe*⟩, così che la sintassi del comando `\documentclass` possa essere espressa così:

`\documentclass[`⟨*opzioni*⟩`]{`⟨*classe*⟩`}`

Con questo tipo di grammatica gli argomenti che obbligatoriamente devono essere forniti ad un comando che richieda argomenti sono sempre racchiusi fra parentesi graffe, mentre gli argomenti facoltativi, opzionali, sono sempre racchiusi fra parentesi quadre. Le parentesi acute presenti nella descrizione della grammatica servono solo per questa descrizione e non vanno effettivamente introdotte quando il comando viene usato.

4.2 L'inizio del file sorgente

Il comando `\documentclass` è il primo che deve essere dato all'inizio di un file (o nel primo file di un gruppo) da trattare con LaTeX.

La ⟨*classe*⟩ indica il tipo di documento che si intende comporre; le classi standard del sistema sono *book*, per comporre libri, *report* per comporre rapporti 'tecnici', e *article* per comporre articoli; esistono ovviamente altre classi non standard, ma queste tre sono le più note sotto ogni aspetto.

Per questo libro è stato usato il comando

```
\documentclass[a4paper]{book}
```

e con questa dichiarazione iniziale si è detto che si vuole comporre un libro e che lo si vuole stampare su carta ISO-UNI A4.

L'inizio del file sorgente prosegue facoltativamente con una serie di specificazioni per rendere più agevole la scrittura del file sorgente, per scegliere la lingua o le lingue da usare nel documento, per scegliere la codifica di default dei font da usare per comporre il documento, per estendere i comandi di default con istruzioni non standard al fine di ottenere risultati particolari.

Questo insieme di dichiarazioni preliminari si chiama *preambolo*; non è necessario che il preambolo contenga qualche cosa, ma in pratica il preambolo in Italia e per i documenti in italiano contiene almeno la specificazione dell'uso della lingua italiana e probabilmente la specifica dei font a 256 caratteri che contengono le lettere accentate; questo vale praticamente in ogni paese, tranne quando la lingua da usare è solamente l'inglese che per LaTeX è la lingua di default e la sua scrittura non contiene lettere accentate. Per questo libro la prima parte del preambolo comincia (nella versione 0.0 cominciava) così:

```
\documentclass[a4paper]{book}
\usepackage[italian]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{guit}

\author{\GuIT}
\title{Introduzione all'arte\\
          della composizione tipografica con~\LaTeX}
\date{Versione 0.0}
```

Come vi si può leggere in chiaro, in questo preambolo si ordina a LaTeX di usare un certo numero di pacchetti di estensione mediante il comando `\usepackage`: in ordine gli si comanda di usare il pacchetto **babel** per gestire lingue diverse dall'inglese e come opzione gli si specifica l'italiano; poi gli si ordina di usare il pacchetto per gestire l'immissione del testo, specificandogli come opzione che i caratteri immessi corrispondono alla codifica *latin1*; poi gli si ordina di usare il pacchetto per gestire la codifica (encoding) per i font di composizione del documento mediante la sigla *T1*, che corrisponde alla codifica standard di LaTeX per poter usare i caratteri latini accentati; infine gli si ordina di usare il pacchetto **guit** per poter avere a disposizione il comando `\GuIT` per comporre la sigla del Gruppo degli Utilizzatori Italiani di TeX; volendo si potrebbe usare anche il logo del gruppo.

Dopo la riga vuota (o bianca) si specifica che l'autore di questo libro è il GuIT stesso, si specifica il titolo del libro e, invece della data, si specifica il numero della versione. 0.0 è il valore iniziale, ma è probabile che il documento che state leggendo abbia un numero successivo, che non sia, cioè, la versione 'zeresima'.

Il preambolo di questo libro contiene altre cose che verranno descritte più avanti. Finito il preambolo comincia la composizione vera e propria del documento.

4.3 Il documento

Il documento comincia prevedibilmente con

```
\begin{document}
```

e, sempre prevedibilmente, finisce con

```
\end{document}
```

Tutto ciò che è racchiuso fra queste due righe viene composto o viene usato per controllare la composizione. È opportuno notare che tutto ciò che nel file compare dopo `\end{document}` non viene stampato ma può essere usato per scrivere dei commenti relativi al file, allo stato di avanzamento della composizione del documento, o ad altro che possa interessare il compositore per sua futura memoria.

In questo libro il documento comincia (nella versione 0.0 cominciava) così:

```
\begin{document}
\frontmatter
\maketitle
\tableofcontents

\chapter*{Presentazione}
```

dove il primo comando dichiara che si stanno componendo le pagine preliminari; il secondo comando serve per comporre la pagina del titolo usando le scarne informazioni circa l'autore, il titolo e la data fornite nel preambolo; il terzo comando ordina di comporre l'indice generale; la riga bianca non conta niente, ma ha un effetto visivo di separazione nel file sorgente; in generale essa serve per marcare la fine di un capoverso, ma in questo punto non ci sono capoversi di nessun genere da comporre.

Il comando `\chapter*` serve per comporre un capitolo non numerato; l'asterisco indica che non si vuole la numerazione; i capitoli non numerati non compaiono nell'indice a meno che non lo si voglia espressamente indicare mediante comandi appositi che per ora esulano da questa descrizione. L'argomento di questo comando indica il titolo da dare a questo capitolo non numerato.

Come si vede il mark-up contiene un'unica piccola difficoltà per coloro che non hanno nessun rudimento di inglese, cioè che tutti i comandi sono in inglese; il buono è che, non contenendo abbreviazioni, questi comandi sono immediatamente comprensibili senza bisogno di ricorrere a elenchi di comandi criptici.

4.4 La fine del documento

Quando il testo del documento è stato completamente introdotto nel file sorgente, non resta che dichiararne la fine chiudendo l'ambiente *document* mediante il comando

```
\end{document}
```

Dopo questa riga e questa istruzione, potete scrivere quello che volete, ma non verrà mai composto; questa è la dichiarazione che il documento è terminato, non che il file è terminato.

4.5 Un semplice esercizio

Esercizio 4.1 Copiate dal testo precedente i comandi illustrati a partire dal comando `\documentclass`; dopo `\chapter*{...}` introducete un qualunque testo con capoversi sufficientemente lunghi, magari un capoverso in italiano, uno in inglese, uno in francese (non dovrebbe essere troppo difficile trovare dei libri o dei giornali che contengano delle frasi, anche in lingua straniera, da copiare, se la propria fantasia e la propria conoscenza delle lingue straniere non fosse sufficiente). Alla fine del documento ricordatevi di dichiararne la fine; sembra lapalissiano, ma talvolta ce ne dimentichiamo.

Salvate il file scritto rigorosamente in formato testuale (meglio se facendo uso di uno shell editor adatto all'uso di LaTeX) per esempio con il nome `esempio1.tex` (il nome non è importante, ma l'estensione *deve* essere `.tex` (o, se proprio volete, `.ltx`).

Lanciate LaTeX; se state lavorando con uno shell editor, cliccate sull'opportuna icona; se non state lavorando con un shell editor aprite la finestra comandi o terminal o xterm o console, comunque si chiami sul vostro calcolatore quella finestra nella quale potete inserire i cosiddetti comandi in linea, scrivendo esplicitamente i nomi dei programmi da lanciare e del file su cui devono operare; in questo esempio scriverete

```
latex esempio1
```

senza specificare l'estensione `.tex` se aveste usato questa estensione, oppure specificando qualunque altra estensione diversa da `.tex`.

Se nella finestra comandi o nell'opportuna finestra dello shell editor compaiono messaggi d'errore, rivedete il testo perché gli unici errori che potete avere fatto consisterebbero in errori di battitura; rivedete con particolare attenzione di aver usato le parentesi graffe dove ci vogliono le graffe, le parentesi quadre dove ci vogliono le quadre, e che tutte le parentesi si accoppino correttamente, cioè che ad ogni parentesi aperta segua una parentesi chiusa dello stesso tipo, come in matematica; quindi: aperta–chiusa, aperta–chiusa,... oppure: aperta, aperta – chiusa, chiusa. Questo è uno degli errori più comuni; inoltre i comandi devono essere scritti correttamente: rivedetene l'ortografia ricordandovi che il correttore ortografico del vostro editor di solito non rileva e non corregge gli errori di battitura dei comandi.

Se le ultime righe di quel che compare nella finestra dicono che il file è stato correttamente salvato nel file `esempio1.dvi`[1] lanciate il previewer adatto a quel formato; potrà essere YAP se avete il sistema MiKTeX; sarà xdvi se state operando su Linux, eccetera.

Ammirate la vostra composizione sullo schermo e osservate se il testo in inglese e/o in francese è stato composto bene, in particolare se le cesure in fin di riga sono corrette.

[1] Se state usando TeXShop su un Mac, il file viene salvato in formato PDF e il previewer viene aperto automaticamente.

Questo semplice esercizio vi permette di rilevare quanto sia semplice usare LaTeX, specialmente se state usando uno shell editor; con quest'ultimo potete premere un solo pulsante che vi attiva l'esecuzione di LaTeX e, se non ci sono errori, vi attiva anche il previewer; ad ogni successiva compilazione generalmente la finestra del previewer viene automaticamente aggiornata, per cui avete una compilazione quasi sincrona; vi consiglio di compilare sovente e di verificare sovente sul previewer che non ci siano errori grossolani di composizione, per esempio potreste aver dimenticato l'effetto di una dichiarazione e quindi potreste scoprire che da un certo punto in avanti state componendo con un font, perché vi siete scordati di ripristinare il font precedente. Questi sono errori grossolani che conviene correggere prima di avere composto le settecento pagine del libro che volete scrivere...

Gli errori più fini vanno ricercati nelle successive revisioni delle bozze, che, per fortuna, non richiedono di stampare su carta una serie di bozze dietro l'altra, ma possono essere riviste direttamente sullo schermo.

Va notato che molti sistemi TeX permettono di eseguire sia la *forward search* sia la *reverse search*. Queste operazioni prevedono una interazione fra l'editor e il previewer in modo che azionando opportuni pulsanti dell'editor mentre il cursore si trova in una certa posizione del testo del file sorgente, nella finestra del file compilato il cursore viene collocato se non nella stessa posizione, qualche parola più avanti o più indietro; questa è la *forward search*. Analogamente con il cursore nella finestra del previewer in una certa posizione, cliccando su un apposito pulsante o premendo una opportuna combinazione di tasti, il cursore si sposta nella finestra dell'editor praticamente nella stessa posizione, parola più, parola meno; questa è la *reverse search*. Grazie a questa interazione fra editor e previewer la correzione delle bozze procede spedita in quanto la finestra dell'editor e quella del previewer sono praticamente sincronizzate.

È probabile che le parti di testo scritte in lingue diverse dall'italiano risultino avere alcune parole divise in sillabe in fin di riga (la cesura) in modo errato; questo dipende dal fatto che nel preambolo dell'esempio si è invocato il pacchetto per il trattamento delle lingue e gli si è specificato di voler comporre in italiano. È ovvio che la sillabazione delle altre lingue è diversa da quella dell'italiano, ed è altrettanto ovvio che scrivendo in lingue diverse dall'italiano è possibile che alcune cesure risultino errate.

Avete probabilmente usato, specialmente per il francese, molte lettere accentate; avrete quindi sfruttato a fondo le possibilità del vostro editor per inserire i caratteri accentati diversi da quelli che vi compaiono già sulla vostra tastiera, come per esempio 'ô' di 'diplôme'; molto probabilmente il file composto visualizzato nel previewer vi mostrerà il segno giusto; se state lavorando con una macchina Windows questo fatto potrebbe non succedere. Per ovviare all'inconveniente sostituite la dichiarazione della codifica *latin1* con la dichiarazione alternativa *ansinew*.

4.6 I caratteri speciali

Per scrivere il mark-up secondo LaTeX bisogna ricordare che alcuni caratteri hanno un significato speciale; essi sono i seguenti:

```
\   {   }   %   $   ^   _   &   #   ~
```

Il carattere \, chiamato *backslash* in inglese e *barra inversa* in italiano, serve come iniziatore di un comando o istruzione, come lo si preferisce chiamare; una istruzione è sempre costituita dal segno \ seguito senza spazi interposti da una *stringa di lettere minuscole e/o maiuscole* e questa stringa termina quando è seguita da un carattere non alfabetico; oppure una istruzione è formata dal carattere \ seguito da *un solo carattere non alfabetico*; oppure essa è costituita da un carattere *attivo*. Nella lista di caratteri speciali illustrata sopra il carattere ~ è un carattere attivo, ed è l'unico carattere attivo definito da LaTeX. Se ne possono avere altri, ma questo si può ottenere solo con pacchetti di estensione; per esempio con l'opzione per la lingua italiana di **babel**, il carattere " è anch'esso attivo, come lo è per quasi tutte le lingue con la notevole eccezione dell'inglese.

I caratteri { e } sono molto particolari perché servono per delimitare gli argomenti dei comandi, ma svolgono anche altre funzioni di cui si parlerà in seguito.

Il carattere % svolge il compito di inizio di un commento: come in tutti i linguaggi di programmazione esiste una maniera per inserire dei commenti in linea con il codice, così anche con il sistema TeX l'inserzione del carattere % in una riga ordina al motore TeX di trascurare tutti i caratteri che seguono % sulla stessa riga, fino al primo carattere diverso da uno spazio (noto anche con il nome di *blank*) nella riga successiva.

Il carattere $ serve come 'interruttore' per ordinare a TeX di passare dal modo testo al modo matematico o viceversa. Quando l'interprete TeX è in modo testo interpreta i caratteri del file d'entrata in una certa maniera, mentre quando è in modo matematico, li interpreta come segni matematici. Per esempio, lo stesso carattere - in modo testo indica un trattino, mentre in modo matematico rappresenta il segno 'meno'; un analogo diverso modo di interpretare i caratteri del file di entrata viene usato da TeX per gestirli appropriatamente quando esso è in modo matematico.

Il segno ^ viene usato solo in matematica e indica che quanto segue eventualmente racchiuso fra parentesi graffe, rappresenta un esponente o un apice. Per usarlo come accento circonflesso nel testo basta farlo precedere da un segno \; per indicare un accento circonflesso in modo matematico bisogna usare il comando `\hat`.

Il segno _ serve in matematica per indicare un deponente o un pedice. Per usarlo come 'underscore' sia nel testo sia in matematica basta farlo precedere dal segno \.

Il segno & serve per separare fra di loro i brevi testi o le brevi espressioni matematiche che entrano nelle celle di una tabella o di una matrice. Preceduto dal backslash serve per comporre l''e commerciale'.

Infine il segno # serve per indicare il numero progressivo dell'elenco dei parametri nella definizione di una nuova istruzione; preceduto dal backslash serve per comporre il *number sign*, in italiano chiamato gergalmente 'cancelletto'.

Per scrivere questi caratteri in un brano di testo senza che svolgano il loro compito normale, bisogna farli precedere dal segno \. Ecco quindi che per esprimere un ammontare di denaro in dollari bisogna usare `\$` nel file sorgente per cui `\$234,99` diventa $234,99. Analogamente per indicare un incremento del 10% bisogna scrivere `10\%` nel file sorgente. L'unico carattere che non è stampabile ricorrendo al segno \ è proprio questo stesso segno \. Per stamparlo bisogna ricorrere al comando `\textbackslash`. Siccome nel testo questo segno

non si usa quasi mai, il fatto di dover ricorrere ad un comando così lungo non costituisce un vero problema.

4.7 Organizzazione dei file sorgente

Questo paragrafo andrebbe scritto in fondo a questo testo, ma tutto sommato è logicamente connesso con il materiale esposto in questo capitolo. All'occorrenza potete saltarne la lettura, salvo poi tornare a questo paragrafo quando dovrete gestire la composizione di un lungo testo molto strutturato.

I semplici esercizi che si possono fare ora e con il contenuto dei primi capitoli che seguono, comportano la scrittura di brevi file sorgente che raramente, una volta compilati, superano il paio di pagine di testo composto.

Il file sorgente non richiede, quindi, una particolare organizzazione, perché la sua suddivisione naturale in 'preambolo' e 'corpo del documento' è più che sufficiente.

Tuttavia si immagini di dover scrivere un testo diviso in numerosi capitoli, appendici, indici e glossari, prefazioni e postfazioni, eccetera. Il file sorgente monolitico rischierebbe di diventare enorme, senza contare l'organizzazione dei file da includere, come i file che contengono le immagini, o i file che conviene predisporre a parte perché richiedono una paziente messa a punto del contenuto.

In questo caso l'organizzazione monolitica non è consigliabile, e infatti LaTeX offre diversi strumenti per spezzare il file sorgente in file di minori dimensioni. LaTeX, infatti, offre due insiemi di comandi per questo scopo: `\input` e `\include` assieme a `\includeonly`, oltre al comando `\includegraphics` per immettere nel flusso di informazioni da elaborare i file contenenti le immagini. Per quest'ultimo comando si veda più avanti il capitolo 8. Per gli altri comandi la sintassi è la seguente:

```
\input{⟨file⟩}

\includeonly{⟨lista di file⟩}
\include{⟨file⟩}
```

Il comando `\input` ordina a LaTeX di leggere il ⟨*file*⟩ specificato; il flusso di dati che viene letto, viene inserito nel flusso complessivo del documento come se fosse inserito esattamente al posto di `\input`; che questo ⟨*file*⟩ contenga una sola parola, o un intero paragrafo o capitolo non ha nessuna importanza; il suo contenuto viene letto incondizionatamente ed elaborato immediatamente.

Questo comportamento del comando `\input` si mantiene anche se è contenuto dentro altri file a loro volta inseriti nel flusso di elaborazione dati mediante altri comandi `\input` o anche comandi `\include`; non esiste un limite teorico al numero di file che possono essere inclusi richiamandosi in catena, salvo la limitazione ovvia delle dimensioni finite della macchina di elaborazione e del numero di canali di ingresso che possono essere aperti contemporaneamente; questo numero massimo di canali è di sedici unità, ma alcuni di questi canali sono già impegnati per altre funzioni di servizio, per cui si può contare su una decina di canali effettivamente disponibili; questo numero dovrebbe essere sovrabbondante rispetto a qualunque esigenza insolita di immissioni a catena mediante comandi `\input`.

Il gioco di `\includeonly` e `\include` è invece più complesso; innanzi tutto, in assenza di `\includeonly` e della sua ⟨*lista di file*⟩, `\include` agisce quasi come `\input`; l'unica differenza è che prima di elaborare il contenuto del file incluso, LaTeX ordina di eseguire un salto di pagina, cioè di iniziare una nuova pagina. Questo permette di capire subito che `\include` non è usabile al posto di `\input`, proprio a causa di questo salto di pagina. Il comando `\include` è più adatto per includere un intero capitolo (che comincia sempre su una pagina nuova) piuttosto che per immettere semplicemente del testo da elaborare nel punto in cui compare il comando.

Ma tutto diventa più interessante e utile se si fanno agire `\inlcudeonly` ed `\include` assieme; infatti in questo modo `\include` include solo i ⟨*file*⟩ che compaiono nella ⟨*lista di file*⟩; però, se l'argomento di `\include` è un ⟨*file*⟩ che non compare nella ⟨*lista di file*⟩, `\include` ordina a LaTeX di leggere solo il file `.aux`, cioè il file ⟨*file*⟩`.aux`, che contiene le informazioni relative ai riferimenti incrociati e al contenuto dei vari contatori che interessano LaTeX, cosicché anche se il file ⟨*file*⟩`.tex` non viene elaborato, le sue informazioni utili sono tenute in conto e tutto procede agli effetti esterni come se il file fosse stato elaborato. Va da sé che se questo file ⟨*file*⟩`.tex` non è mai stato elaborato prima, non esiste nemmeno il file ⟨*file*⟩`.aux`, ma `\include` non protesta perché si limita a scrivere un avvertimento nel file `.log` privo di qualunque conseguenza sul buon fine della compilazione.

Un file incluso mediante `\include` può contenere a sua volta comandi `\input`, i file dei quali a loro volta... Va da sé però che, se un file non viene incluso perché il suo nome non compare nella ⟨*lista di file*⟩, non vengono elaborati nemmeno gli eventuali file subalterni richiamati dai comandi `\input`.

Al contrario di `\input`, i comandi `\include` non possono essere innestati l'uno dentro l'altro ed è del tutto logico che sia così.

Tutto questo può sembrare oltremodo complicato, mentre un esempio permette di rendersi conto che non solo è semplicissimo, ma è anche molto logico. Si pensi ad un lungo saggio che si ritiene di suddividere nei file seguenti:

1. `premesse.tex` che contiene le informazioni per comporre tutto il materiale iniziale, dall'occhiello fino alla prefazione, inclusi tutti gli indici che si vogliono creare.
2. `introduzione.tex` che contiene un capitolo introduttivo.
3. `statodellarte.tex` che contiene una dettagliata descrizione della situazione pertinente prima della teoria esposta nel saggio.
4. `motivazioni.tex` dove si descrivono le motivazioni che portano allo sviluppo della nuova teoria descritta nel saggio.
5. `sviluppo.tex` dove si discute lo sviluppo della nuova teoria.
6. `risultati.tex` dove si discutono i risultati alla luce della nuova teoria.
7. `confronti.tex` dove si discutono e si confrontano i nuovi risultati con quelli che si potevano ottenere applicando le teorie precedenti. Magari questo file contiene anche le considerazioni conclusive.
8. `appendici.tex` dove si raccolgono tutte le informazioni, dalla bibliografia alle informazioni accessorie allo sviluppo della nuova teoria, fino, eventualmente, agli indici analitici e ai glossari.

Si procederà allora a predisporre un 'master file' `nuovateoria.tex` nel quale si mettono solo il preambolo e i comandi per l'inclusione dei vari file:

```
% file nuovateoria.tex
%
\documentclass[⟨opzioni⟩]{⟨classe⟩}
⟨preambolo che contiene tutti i comandi per i pacchetti da usare, le definizioni
di nuovi comandi, e si conclude con:⟩
%
\includeonly{%
premesse,%
introduzione,%
statodellarte,%
motivazioni,%
sviluppo,%
risultati,%
confronti,%
appendici%
}
%
% Qui finisce il preambolo e comincia il documento
%
\begin{document}

\include{premesse}
\mainmatter
\include{introduzione}
\include{statodellarte}
\include{motivazioni}
\include{sviluppo}
\include{risultati}
\include{confronti}
\backmatter
\include{appendici}

\end{document}
```

Se si elabora il file `nuovateoria.tex` così come è presentato sopra, LaTeX legge, se ci sono, tutti i file dichiarati come argomento ai vari comandi `\include`. Ma si osservi come è scritta la ⟨*lista di file*⟩ che costituisce l'argomento di `\includeonly`; ogni file della lista è scritto all'inizio di una nuova riga la quale è terminata dal segno di commento al fine di non lasciare spazi spuri nella lista. In questo modo c'è la possibilità di inserire un segno di commento all'inizio delle righe che contengono i nomi dei file che non si desiderano elaborare.

All'inizio della scrittura dell'intero documento si comincerà presumibilmente con il file `premesse.tex`; si metterà allora un segno di commento davanti al nome di tutti gli altri file; quando poi si passa a scrivere `introduzione.tex`, si toglie il segno di commento davanti al nome di questo file e lo si inserisce, invece, davanti al nome di `premesse.tex`. Via via che si procede, si compone un file alla volta e se ne controlla il risultato attraverso il programma di visualizzazione adeguato al tipo del file di uscita. Ogni volta la compilazione ha luogo su un solo file ed

è particolarmente veloce; nello stesso tempo i riferimenti incrociati a elementi contenuti nei capitoli precedenti vengono tutti risolti correttamente. Anche le bozze, se le si volesse stampare, hanno le pagine numerate correttamente. Solo alla fine, a lavoro quasi terminato, si tolgono tutti i segni di commento davanti ai nomi dei file della ⟨*lista di file*⟩, e si esegue la composizione dell'intero testo in un unico file di uscita.

Se lo shell editor lo richiede (e quasi tutti gli shell editor lo richiedono) bisogna solo specificare all'editor il nome del 'master file' affinché, quando si compila e l'editor ha aperto solo uno dei file da includere, esso possa lanciare LaTeX sul 'master file', e non sul file correntemente aperto.

Questo tipo di organizzazione del file d'entrata diventa più comodo da usare via via che si acquisisce pratica e se ne apprezzano i vantaggi.

Vale la pena di sottolineare che la divisione in capitoli da includere con il comando `\include` va benissimo con i capitoli. Il comando `\input` serve per gli scopi indicati, ma l'esperienza insegna a servirsene con il massimo profitto in diverse altre circostanze; per esempio lo scrivente ha sviluppato l'abitudine di comporre grandi tabelle in file a se stanti da includere con `\input`. Questo modo di procedere deriva dalla constatazione che le grandi tabelle sono complesse da comporre ed è facile commettere errori. Inserite in file a se stanti, e creato un unico piccolo 'master file' che ordini la lettura solo del file contenente la tabella, ogni tabella viene composta e corretta per suo conto e non c'è pericolo che gravi errori, particolarmente difficili da individuare, fermino la compilazione di un grosso file in un punto che non ha nulla a che vedere con la posizione dell'errore, come, ahimè, talvolta succede.

Dalla versione 0.09 anche i file che compongono questo testo sono gestiti con il meccanismo di `\include` e `\incldeonly`. Quest'ultimo, collocato nel preambolo, è formato così:

```
\includeonly{%
preliminari,%
presentazione,%
sincrona-asincrona,%
nozioni-tipografia,%
installazione,%
prime-nozioni,%
testi-speciali,%
tabelle,%
figure,%
importazione-figure,%
matematica1,%
matematica2,%
caratteri,%
presentazioni,%
classi,%
bibtex,%
indici-glossari,%
nuovi-comandi,%
documentazione,%
filologia,%
simbologia,%
divisione-sillabe,%
```

```
sintassi-riepilogo,%
indanalitico%
}
```

e, come si vede, ogni capitolo, numerato o non numerato, e ogni appendice viene elencata nella ⟨*lista di file*⟩ nella speciale maniera descritta sopra. Successivamente il corpo del testo è formato dal blocco:

```
\begin{document}
%
\frontmatter
%
\include{preliminari}
\include{presentazione}
%
\mainmatter
%
\include{sincrona-asincrona}
\include{nozioni-tipografia}
\include{installazione}
\include{prime-nozioni}
\include{testi-speciali}
\include{tabelle}
\include{figure}
\include{importazione-figure}
\include{matematica1}
\include{matematica2}
\include{caratteri}
\include{presentazioni}
\include{classi}
\include{bibtex}
\include{indici-glossari}
\include{nuovi-comandi}
%
\appendix
%
\include{documentazione}
\include{filologia}
\include{simbologia}
\include{divisione-sillabe}
\include{sintassi-riepilogo}
\include{indanalitico}
%
\end{document}
```

I comandi `\frontmatter` e `\mainmatter` specificano l'inizio dei preliminari e l'inizio del corpo del testo e sono presenti anche se gli eventuali file che dovrebbero venire inclusi di fatto non lo sono perché eliminati con segni di commento dalla lista dei file di `\includeonly`; invece di `\backmatter` si è usata solo la specifica `\appendix` che ordina che da quel punto in poi i capitoli sono da trattare come appendici; non si è usato `\backmatter` perché chi ha predisposto la classe *`book`*

ha ritenuto che le appendici facessero parte del corpo del testo e non delle parti finali.

4.8 Gestione degli errori

Purtroppo talvolta i file sorgente contengono errori; in queste circostanze l'interprete TeX procede ad interpretare il flusso di informazioni che via via legge dai file sorgente, finché riesce a dare loro un qualche significato, poi si ferma con un messaggio d'errore. Quasi sempre si tratta di un comando scritto con un errore di ortografia; sullo schermo appare allora il messaggio d'errore con una qualche frase esplicativa e la linea dove TeX si è bloccato scritta 'spezzata' in corrispondenza dell'ultimo oggetto che TeX ha letto. Se si tratta di un errore di ortografia in un comando, appare appunto il comando errato alla fine del primo moncone della riga presentata e la riga del flusso d'entrata non ancora sottoposto al processo di interpretazione appare subito sotto. Il primo moncone di riga è anche preceduto dal numero della riga del file sorgente.

Le risposte possibili ad un messaggio d'errore, sono quelle di introdurre una delle seguenti lettere: `x`, `h`, `i`, `e`, eventualmente seguita da un stringa di testo.

`x` serve per terminare subito la compilazione; successivamente si può dare il comando allo shell editor di collocare il cursore nella riga contenente l'errore in modo da poterlo correggere e riprendere la compilazione.

`h` serve per chiedere aiuto; i messaggi informativi che LaTeX fornisce sullo schermo talvolta permettono di capire di che errore si tratti; talvolta sono troppo generici e bisogna armarsi di pazienza e di ragionamento investigativo per trovare che cosa c'è che non va.

`e` serve per uscire dal programma di compilazione e di ritornare immediatamente allo shell editor con il cursore già messo nella riga con l'errore; questa operazione ha esito felice se c'è una perfetta integrazione fra compilatore e shell editor; ma talvolta non riesce perché lo shell editor non è configurato correttamente.

`i` serve per introdurre una stringa di testo *al posto* dell'ultimo oggetto letto dall'interprete; per esempio se si scrivesse `\iput`, TeX si fermerebbe dopo aver letto questo oggetto a cui non sa dare un significato, perché è un comando scritto male; al prompt di TeX (costituito da un punto interrogativo) basta scrivere `i\input` e poi premere il tasto di invio; TeX sostituisce `\input` all'oggetto ignoto `\iput` e procede nella compilazione; si noti che il file sorgente non è stato toccato da questa operazione, quindi l'errore esiste ancora e bisogna correggerlo; bisogna cioè ritornare allo shell editor e chiedergli di collocarsi alla riga che contiene l'errore.

Se si ha l'accortezza di lanciare LaTeX abbastanza sovente, ogni possibile errore, anche difficilissimo da trovare, può trovarsi solo nel testo sorgente introdotto fra la penultima e l'ultima volta che si è eseguita la compilazione; se questa viene eseguita sovente, il testo introdotto è relativamente corto e quindi risulta più facile trovare l'errore.

Capitolo 5

LaTeX: testi speciali

I testi speciali sono quelli in display e le note.

5.1 Che cosa sono i testi in display

I testi in display sono quelli 'in vetrina', quelli messi in evidenza in qualche modo o per il loro contenuto o per quello che rappresentano; si distinguono:

- le citazioni
- gli elenchi
- le descrizioni
- le liste bibliografiche

e verranno ora descritti una categoria alla volta.

5.2 Le citazioni

LaTeX offre tre ambienti per mettere in evidenza dei testi citati;

1. le citazioni brevi
2. le citazioni lunghe
3. le poesie

5.2.1 Le citazioni brevi

Una citazione di un testo altrui che consista in poche parole può essere eseguita all'interno del testo racchiudendola fra virgolette. Le virgolette si indicano con i segni `` per le virgolette aperte e con '' per le virgolette chiuse. Siccome la tastiera italiana manca del tasto per segnare l'accento grave, bisogna appoggiarsi alle potenzialità dello shell editor per inserire questi segni mancanti, oppure bisogna conoscere i codici immissibili nel file d'entrata mediante apposite combinazioni di tasti. Le virgolette caporali sono più facili da immettere quando si usa la codifica `T1` per il font di composizione del documento; basta scrivere

`<<` per i caporali aperti e `>>` per i caporali chiusi; si ottiene « ». Solo che i caporali non sono adatti per le citazioni ma sono usati più frequentemente per i dialoghi[1].

A questo proposito va segnalato che le virgolette aperte “ e quelle chiuse ” sono asimmetriche; in tipografia non bisogna mai usare le doppie virgolette simmetriche, anche se noti word processor lo fanno. Per questo motivo non bisogna mai usare il segno `"` per inserire le virgolette. Anzi con l'opzione per l'italiano di **babel** il segno `"` è attivo e quindi si comporta come una istruzione. Precisamente se esso è seguito da un qualunque carattere alfabetico esso inserisce un punto di possibile cesura e questa possibilità torna molto comoda per indicare dove andare a capo in quelle parole composte dove si vuole evitare la sillabazione fonetica prescritta dalle grammatiche e dalle norme tipografiche, ma si vuole usare la sillabazione etimologica: se si scrive `macro"istruzione` le due parole `macro` e `istruzione` sono trattate separatamente ai fini della sillabazione, ma una eventuale cesura viene esplicitata all'occorrenza anche fra la 'o' e la 'i' se la cesura in fin di riga lo richiede. Allo stesso modo `"/` consente di andare a capo dopo la barra quando si scrive, per esempio, `modulazione"/demodulazione` che viene composto come modulazione/demodulazione; in fin di riga modulazione/ demodulazione.

Tornando alle virgolette, siccome con la tastiera italiana l'inserzione delle virgolette aperte è problematica, se si usa **babel** con l'opzione per l'italiano, basta scrivere `""` per ottenere “.[2]

5.2.2 Le citazioni lunghe

Le citazioni più lunghe, anche consistenti in qualche riga di testo ma complessivamente senza superare la lunghezza di un capoverso, possono essere evidenziate mediante l'ambiente *quote* in questo modo:

[1] Con l'opzione *italian* di **babel**, indipendentemente dalla codifica usata per il testo di uscita, si può ricorrere alle funzionalità del carattere attivo `"` e scrivere `"<` per i caporali aperti e `">` per i caporali chiusi. Inoltre si può scrivere `""` per sfruttare le potenzialità del carattere attivo `"` che introduce in questo caso le doppie virgolette alte aperte “ come se si fossero potuti introdurre da tastiera due accenti gravi in sequenza.

[2] Con la tastiera italiana è generalmente difficile introdurre i segni ASCII che i programmatori che hanno stabilito le corrispondenze fra i tasti e i segni da scrivere hanno volutamente o inconsapevolmente trascurato, pensando che i PC sarebbero serviti principalmente per il lavoro d'ufficio, e non per programmare; i segni vistosamente mancanti in relazione all'uso di LaTeX sono `, ~, {, }. Con le varie incarnazioni di Windows per ottenere i primi due segni bisogna usare il tastierino numerico e premere i tasti corrispondenti ai codici 96 o 126 rispettivamente, mentre si tiene premuto il tasto Alt di sinistra. I secondi due segni, le parentesi graffe, benché non sia segnato sui tasti, si trovano sulle maiuscole delle parentesi quadre, le quali si ottengono premendo il tasto AltGr insieme al tasto contenente il disegno della parentesi quadra desiderata; se si preme anche il tasto delle maiuscole si scrive la corrispondente parentesi graffa. Con Linux c'è la possibilità di configurare la tastiera sfruttando i tasti Alt e AltGr, oltre a Ctrl e Shift, in modo da ottenere una grande varietà di segni. Con Mac OS X esiste la possibilità di aprire il *Keyboard viewer* e trovare visivamente quali combinazioni di tasti producano quali segni. In generale sarebbe preferibile usare sempre una tastiera con il layout dei tasti corrispondente alla versione USA e poi usare le comode personalizzazioni per scrivere le lettere accentate, mancanti dalla tastiera USA, ma sapendo che la stessa tecnica serve anche per le lettere maiuscole e per inserire gli accenti che non si usano in italiano, ma sono necessari per scrivere in altre lingue, anche solo poche parole.

```
\begin{quote}
⟨breve testo da citare⟩
\end{quote}
```

Il testo viene composto mettendolo in evidenza fra margini ristretti rispetto al testo normale.

Per le citazioni più lunghe si usa l'ambiente *quotation*; la sintassi è:

```
\begin{quotation}
⟨testo lungo da citare⟩
\end{quotation}
```

La differenza rispetto all'ambiente *quote* è che in questo caso la citazione può contenere diversi capoversi; questi a loro volta, oltre ad essere composti fra margini ristretti, vengono composti con la prima riga rientrata del solito rientro di capoverso.

I versi sono una forma particolare di citazione, nel senso che viene composto un testo scritto da altri; tuttavia si tratta di un testo strutturato in versi e strofe; l'ambiente *verse* richiede che alla fine di ogni verso sia indicato l'ordine di andare a capo mediante il comando \\ e che ogni strofa sia trattata come un capoverso, cioè che venga lasciata una riga bianca fra una strofa e la successiva.

La sintassi è la seguente:

```
\begin{verse}
⟨verso⟩\\
⟨verso⟩\\
...
⟨verso⟩

⟨verso⟩\\
⟨verso⟩\\
...
⟨verso⟩
\end{verse}
```

Ecco, per esempio, il famoso sonetto di Dante Alighieri[3]:

> Tanto gentil e tanto onesta pare
> la donna mia quand'ella altrui saluta,
> ch'ogne lingua deven tremando muta,
> e li occhi no l'ardiscon di guardare.
>
> Ella si va, sentendosi laudare,
> benignamente d'umilta' vestuta;
> e par che sia una cosa venuta
> da cielo in terra a miracol mostrare.
>
> Mostrasi si' piacente a chi la mira,
> che da' per li occhi una dolcezza al core,
> che 'ntender non la puo' chi no la prova;

[3]La mancanza di accenti, sostituiti con gli apostrofi di elisione, sembra che sia una caratteristica dei manoscritti più accreditati.

e par che de la sua labbia si mova
uno spirito soave pien d'amore,
che va dicendo a l'anima: Sospira.

viene composto con i comandi seguenti:

```
\begin{verse}
Tanto gentil e tanto onesta pare\\
la donna mia quand'ella altrui saluta,\\
ch'ogne lingua deven tremando muta,\\
e li occhi no l'ardiscon di guardare.

Ella si va, sentendosi laudare,\\
benignamente d'umilta' vestuta;\\
e par che sia una cosa venuta\\
da cielo in terra a miracol mostrare.

Mostrasi si' piacente a chi la mira,\\
che da' per li occhi una dolcezza al core,\\
che 'ntender non la puo' chi no la prova;

e par che de la sua labbia si mova\\
uno spirito soave pien d'amore,\\
che va dicendo a l'anima: Sospira.
\end{verse}
```

Un tipo di citazione o di messa in evidenza di un testo è quello che serve per presentare brani di programmazione, come succede spesso in questo libro. Questo tipo di composizione richiede che nessun carattere sia speciale, che il font usato sia tale da rendere chiaro il fatto che il testo citato è un brano di programmazione, dove gli spazi e le rientranze possono avere un importanza particolare, dove le linee troppo lunghe vanno a capo ma senza dividere in sillabe nessuna parola e dove anche gli 'a capo' devono essere rispettati senza preoccuparsi di giustificare il testo. Questo risultato si ottiene ricorrendo all'ambiente *verbatim*[4]. L'unica cosa che non si può riprodurre è proprio costituito dalla stringa di caratteri `\end{verbatim}` perché questa stringa, nella sua interezza serve per riconoscere la fine dell'ambiente. Questo testo abbonda di esempi di brani di programmazione; il più recente è quello usato per rappresentare la codifica della poesia citata poco sopra. La sintassi dell'ambiente *verbatim* è la seguente:

```
\begin{verbatim}
⟨testo da riprodurre verbatim⟩
\end{verbatim}
```

5.3 Gli elenchi

Gli elenchi sono generalmente di tre tipi: (*a*) in linea con il testo, come questo elenco; (*b*) in display con gli oggetti elencati distinti per mezzo di una numerazione, per cui questo tipo di elencazione si chiama *enumerazione*; e (*c*) in

[4] 'verbatim' in latino significa 'alla lettera'.

display dove però ogni elemento è evidenziato mediante un simbolo grafico, ma questo simbolo non è distintivo del particolare oggetto elencato; per cui questa è una elencazione semplice.

5.3.1 Le elencazioni in linea

Le elencazioni in linea sono quasi sempre composte come l'elencazione precedente, dove il segno distintivo di ogni oggetto elencato è una lettera in corsivo racchiusa fra parentesi tonde. L'elenco non è particolarmente evidenziato e la presenza delle lettere corsive è l'unico elemento che fa capire che si tratta di una elencazione; non è frequente, ma queste lettere corsive possono essere usate per richiamare l'oggetto identificato, nel qual caso devono essere riportate anche le parentesi, come quando ci si vuol riferire al caso (a).

5.3.2 Le enumerazioni

Gli elenchi numerati sono molto più strutturati anche in relazione alla possibilità di annidare un elenco dentro un altro. Le enumerazioni vengono realizzate ricorrendo all'ambiente *enumerate* con la seguente sintassi:

```
\begin{enumerate}
\item[⟨opzione⟩] ⟨oggetto testuale⟩
\item[⟨opzione⟩] ⟨oggetto testuale⟩
...
\end{enumerate}
```

Perciò:

1. L'enumerazione viene composta dentro un ambiente *enumerate* come era facile prevedere;
2. ogni oggetto da enumerare viene introdotto con il comando `\item`;
3. se per questo comando non si specifica nessuna opzione (e quindi non si scrivono nemmeno le parentesi quadre) l'oggetto viene numerato con un 'numero' progressivo;
4. questo 'numero' può essere qualsiasi oggetto appartenente ad un sequenza ordinata di oggetti, tipicamente:
   (a) dei numeri arabi, o
   (b) dei numeri romani, oppure
   (c) delle lettere dell'alfabeto latino minuscolo, o
   (d) delle lettere dell'alfabeto latino maiuscolo.

   La classe del documento specifica quale tipo di numerazione viene eseguito per ogni livello di enumerazione annidata. Di default LaTeX con la sua classe *book* usa i numeri arabi, le lettere minuscole, i numeri romani, le lettere maiuscole.
5. Se invece viene espressa l'opzione, foss'anche una opzione 'nulla', allora invece del 'numero' viene scritta l'opzione, perciò

⋆ si può fare riferimento simbolico agli elementi numerati per davvero, ma non si può fare riferimento, per esempio a questo elemento, perché non è numerato.

6. Come tutte le elencazioni, esse possono essere annidate una dentro l'altra, ma non si possono usare più di tre elencazioni annidate dentro una elencazione principale, non tanto perché la cosa sia tecnicamente impossibile, ma perché la struttura diventerebbe troppo complicata e il lettore ne rimarrebbe confuso.

5.3.3 Le elencazioni semplici

Le elencazioni semplici sono in display come le enumerazioni, ma ogni elemento elencato è solamente contrassegnato con un simbolo grafico uguale per tutti gli elementi dello stesso livello di elencazione. L'ambiente si chiama *itemize* e vale la sintassi seguente:

```
\begin{itemize}
\item[⟨opzione⟩] ⟨oggetto testuale⟩
\item[⟨opzione⟩] ⟨oggetto testuale⟩
...
\end{itemize}
```

Perciò

- il primo livello di elencazione viene contrassegnato con dei pallini neri, ma

⋆ se viene espressa una opzione, come in questo caso, il contrassegno è quanto viene specificato come opzione.

- Le elencazioni semplici possono essere annidate l'una dentro l'altra e il contrassegno cambia secondo il livello di annidamento. per esempio:
  - in questo secondo livello il contrassegno è costituito da un lineato medio;
  - si possono cambiare questi contrassegni agendo sulle definizioni che sono contenute nel file di classe.

- In ogni caso si ricorre a queste elencazioni semplici quando gli oggetti da elencare non sono logicamente sequenziali.

5.3.4 Alcune osservazioni relative alle elencazioni

Nei paragrafi precedenti si sono descritte alcune elencazioni; la domanda ricorrente è la seguente: Quale tipo di punteggiatura viene usato fra un elemento e il successivo?

Per le elencazioni in linea si suppone che gli oggetti elencati siano brevi frasi, costituite al massimo da un solo periodo, quindi il punto e virgola sembra essere il segno di interpunzione più appropriato. Se non fosse così, allora sarebbe meglio costruire una elencazione in display.

Per le elencazioni in display, semplici o numerate, la punteggiatura può ancora essere un punto e virgola se gli oggetti sono brevi frasi, o frazioni di frase;

ma se sono costituiti da periodi completi o da più periodi è necessario usare la punteggiatura che si userebbe anche senza evidenziare l'elencazione. Non necessariamente ogni oggetto elencato deve finire con la stessa punteggiatura, anzi, negli elenchi precedentemente usati a titolo di esempio si sono terminati alcuni oggetti senza punteggiatura ma con una congiunzione.

Nelle enumerazioni è possibile inserire il comando `\label` dentro gli oggetti a cui si vuole fare riferimento in modo simbolico, per esempio si può inserire nell'ultimo elemento dell'enumerazione usata come esempio:

```
\item \label{ele:annidato}Come tutte le elencazioni,
```

così che qui vi si possa fare riferimento mediante l'uso dei comandi `\ref` e `\pageref`, scrivendo

```
l'elemento~\ref{ele:annidato} nella pagina~\pageref{ele:annidato}.
```

ottenendo: "l'elemento 6 nella pagina 40."

5.4 Le descrizioni

Le descrizioni possono essere viste come elencazioni particolari nelle quali il segno distintivo non è né un numero né un simbolo, ma è una parola o una locuzione di cui si fornisce una descrizione o una spiegazione nel resto del testo dell'oggetto; la sintassi dell'ambiente *description* è la seguente:

```
\begin{description}
\item[⟨locuzione⟩] ⟨descrizione⟩
\item[⟨locuzione⟩] ⟨descrizione⟩
...
\end{description}
```

Benché la ⟨*locuzione*⟩ sembri facoltativa, essa in effetti è logicamente necessaria, perché non avrebbe nessun senso dare una ⟨*descrizione*⟩ di... nulla.

Questo ambiente è particolarmente indicato per dare una serie di definizioni, oppure per scrivere un glossario.

Esercizio 5.1 Continuando l'esercizio 4.1 si cominci un nuovo capitolo senza specificare l'asterisco, ma dandogli un titolo a piacere; si componga poi un ambiente *description* nel quale si danno le definizioni di avanzamento di riga, di corpo e di interlinea.

5.5 Le liste bibliografiche

Le liste bibliografiche sono dei particolari elenchi numerati dove i riferimenti bibliografici sono etichettati con dei numeri che possono essere richiamati simbolicamente nel corso del testo. Va subito detto che qui si parla dell?elenco bibliografico standard, cioè quello che si ottiene utilizzando l?ambiente *thebibliography* con la sintassi seguente per comporre l?elenco dei riferimenti bibliografici:

```
\begin{thebibliography}{⟨stringa⟩}
\bibitem[⟨richiamo⟩]{⟨chiave⟩} ⟨riferimento bibliografico⟩
\bibitem[⟨richiamo⟩]{⟨chiave⟩} ⟨riferimento bibliografico⟩
...
\end{thebibliography}
```

dove

⟨*stringa*⟩ è una stringa di caratteri, generalmente una stringa di cifre, ma il significato numerico è irrilevante; serve all'ambiente per prendere le misure per la lunghezza del campo da lasciare sul margine sinistro dell'elenco al fine di incolonnare correttamente tutte le voci della bibliografia.

⟨*richiamo*⟩ è una parola o una stringa di lettere e numeri che permette di identificare il particolare riferimento bibliografico all'interno del testo, oltre che nel campo identificativo dell'elenco; se non si specifica niente, (nemmeno le parentesi quadre), il riferimento di default è costituito da un numero, progressivo all'interno dell'elenco, racchiuso fra parentesi quadre. Questo è il modo di citazione più frequente negli articoli scientifici che si riferiscono a scienze nelle quali si fa un grande uso del linguaggio matematico.

Nelle discipline umanistiche è più frequente il modo di citazione 'autore-anno', per cui il ⟨*richiamo*⟩ è appunto costituito dal cognome del (primo) autore seguito dall'anno di pubblicazione dell'opera citata e, in caso di ambiguità, da una lettera progressiva. Ecco allora che la possibilità di stabilire con precisione il nome dell'autore e l'anno di pubblicazione dell'opera citata diventa un ottimo modo per specificare il ⟨*richiamo*⟩.

⟨*chiave*⟩ è il nome simbolico da usare sia nell'eventuale database bibliografico, sia come argomento obbligatorio del comando `\bibitem`, sia come argomento del comando `\cite`, che serve per citare il lavoro all'interno del documento.

L'ambiente *thebibliography* provvede a iniziare una nuova pagina, a scrivere una intestazione di capitolo non numerato, ma intitolata Bibliography, oppure Bibliografia, oppure..., a seconda della lingua di default.

Se questo ambiente contenesse una voce del genere:

```
\bibitem{Lamp94} \textsc{Leslie Lamport},
\textit{A document preparation system --- \LaTeX\ ---
User's guide and reference manual}, Addison Wesley,
Reading, Mass., 2nd ed., 1994.
```

nella bibliografia apparirebbe una voce:

[12] Leslie Lamport, *A document preparation system — LaTeX — User's guide and reference manual*, Addison Wesley, Reading, Mass., 2nd ed., 1994.

e l'opera citata verrebbe citata con `\cite{Lamp94}`, per ottenere nel testo '[12]'.

Se invece nell'elenco bibliografico si fosse scritto:

```
\bibitem[\textsc{Lamport-94}]{Lamp94} \textsc{Leslie Lamport},
\textit{A document preparation system --- \LaTeX\ ---
User's guide and reference manual}, Addison Wesley,
Reading, Mass., 2nd ed., 1994.
```

nella bibliografia apparirebbe una voce

[LAMPORT-94] LESLIE LAMPORT, *A document preparation system — LaTeX — User's guide and reference manual*, Addison Wesley, Reading, Mass., 2nd ed., 1994.

e l'opera, citata sempre con la stessa chiave `\cite{Lamp94}`, apparirebbe nel testo come [LAMPORT-94].

Facendo ricorso a pacchetti di estensione da richiamare con `\usepackage` nel preambolo, è possibile comporre questi elenchi in modo stilisticamente diverso; non solo, ma è possibile estrarre da un database bibliografico le informazioni necessarie mediante il programma BibTeX, così da avere un risultato compositivamente omogeneo oltre che semanticamente completo.

5.6 I riferimenti incrociati

Come si è visto nei paragrafi precedenti, è possibile riferirsi simbolicamente a qualunque oggetto numerato e alla pagina in cui compare, se questo tipo di informazione è sensato.

I comandi per ottenere questi riferimenti sono i seguenti e soddisfano alla sintassi qui di seguito indicata.

```
\label{⟨chiave⟩}
\ref{⟨chiave⟩}
\pageref{⟨chiave⟩}

\bibitem[⟨riferimento⟩]{⟨chiave⟩}
\cite[⟨informazioni aggiuntive⟩]{⟨chiavi⟩}
```

La ⟨*chiave*⟩ è una etichetta simbolica, possibilmente mnemonica (composta con qualunque sequenza di caratteri esclusa la virgola) con la quale si vuole identificare un oggetto. Come si è visto negli esempi precedenti, io compongo la chiave con una breve sigla che ricorda il tipo di oggetto, per esempio `fig:` per una figura, `ese:` per un esempio, `cap:` per un capitolo, `equ:` per una equazione, eccetera, seguito da un nome mnemonico che mi ricordi di che cosa si tratti.

Il comando `\label` con la sua ⟨*chiave*⟩, va inserito nel file sorgente *dopo* il comando che assegna un numero all'oggetto; i comandi `\ref` e `\pageref` con le loro chiavi vanno scritti nel testo nei punti dove si vuole fare riferimento all'oggetto; una precauzione utile dal punto di vista tipografico è quella di far precedere i comandi `\ref` e `\pageref` da una tilde, che in LaTeX ha il significato di *spazio non separabile*, così che durante la composizione il motore TeX non spezzi la riga o non cambi pagina fra il nome che precede il riferimento e il riferimento stesso. Si scriverà, per esempio, `nell'equazione~\ref{equ:Pitagora}` così che TeX non spezzi la riga fra la parola 'equazione' e il numero che esso sostituirà alla ⟨*chiave*⟩ 'equ:Pitagora'.

Facendo uso del pacchetto di estensione **varioref** (da richiamare mediante il comando `\usepackage` nel preambolo) è possibile avere i riferimenti incrociati completi di numero e pagina in locuzioni del tipo 'la figura 5.13 nella pagina seguente' oppure 'la figura 5.14 nella pagina 215' a seconda della distanza del richiamo dal punto in cui l'oggetto compare nel documento.

Per i riferimenti bibliografici il comando `\bibitem`, come si è visto, consente di definire una chiave e, facoltativamente, un richiamo. Il comando `\cite` accetta non solo una chiave, ma diverse chiavi separate da virgole; si potrebbe, per esempio, scrivere `\cite{lamp94,TexComp}` e ritrovarsi scritto nel testo '[12, 13]', cioè la citazione di entrambe le opere identificate mediante i loro numeri progressivi all'interno dell'elenco bibliografico.

Il comando `\cite` accetta anche un argomento facoltativo (che ha senso quando si cita una sola chiave) per indicare un posto preciso dell'opera citata; per esempio, se si scrive `\cite[cap.~4]{Lamp94}` si ottiene '[12, cap. 4]'.

Il vantaggio che questo metodo di citazione simbolico offre è che in caso di correzioni, per esempio cancellazioni o inserimenti di parte del testo, non è necessario correggere tutte le citazioni, ma ci pensa LaTeX a correggere tutti i valori che risultano modificati; bisogna solo lanciare LaTeX un paio di volte, finché nei messaggi finali non compare più il messaggio che 'alcuni riferimenti potrebbero essere stati modificati'.

5.7 Altri testi in display

Talvolta è necessario mettere in evidenza, separandolo dal resto del testo, un brano composto senza giustificazione, o meglio, solo allineato a sinistra, oppure solo allineato a destra, oppure centrato.

Per questo scopo sono disponibili i tre ambienti *flushleft*, *flushright* e *center*.

Si ricorda che il testo giustificato solo a sinistra è più facilmente leggibile di quello giustificato solo a destra e del testo in stile epigrafico con le righe centrate. Tuttavia per i titoli, per esempio, lo stile con le righe centrate è quasi sempre più indicato che non lo stile giustificato solo da un lato o giustificato da entrambi i lati.

Questi tre ambienti compongono il testo che essi racchiudono nella maniera specifica di ciascuno, ma tutti e tre hanno la caratteristica che il testo così composto risulta distanziato sopra e sotto mediante l'equivalente di una riga bianca (vuota).

In certe circostanze non è questo quello che si desidera, ma si vuole cambiare temporaneamente il tipo di giustificazione senza lasciare spazi bianchi prima o dopo il testo così composto.

Esistono i tre comandi (dichiarazioni) `\raggedright`, `\raggedleft` e, naturalmente, `\centering` che ottengono lo scopo desiderato; siccome non esiste un comando `\justify` che permetta di tornare al testo giustificato da entrambi i lati, è necessario limitare l'effetto di questi tre comandi mediante l'uso di un *gruppo*; un gruppo è una parte di testo intercalato a comandi racchiuso (*a*) fra due parentesi graffe, oppure (*b*) fra i comandi `\bgroup` ed `\egroup`, oppure (*c*) all'interno di ambienti delimitati da `\begin` e `\end`.

Tuttavia bisogna ricordare che l'interprete TeX esegue la composizione di ciascun capoverso, e quindi la sua giustificazione specifica, solo alla fine del capoverso stesso; per indicare esplicitamente la fine di un capoverso all'interno di un gruppo è meglio fare uso del comando `\par` (iniziale di *paragraph*, che in inglese indica il capoverso), che specifica proprio la fine di un capoverso. Perciò scrivere:

```
Andando lungo la strada egli vide una iscrizione
```

```
che annunciava:\\[1ex]
{\centering Qui visse Torquato Tasso durante il suo
soggiorno del 1605}\\[1ex] ma la cosa gli sembrava
strana, poiché ricordava che il Tasso era vissuto  nel~'500.
```

è sbagliato perché produce:

> Andando lungo la strada egli vide una iscrizione che annunciava:
>
> Qui visse Torquato Tasso durante il suo soggiorno del 1605
>
> ma la cosa gli sembrava strana, poiché ricordava che il Tasso era vissuto nel '500.

senza centrare il messaggio lapidario.
Invece scrivendo:

```
Andando lungo la strada egli vide una iscrizione
che annunciava:\par\vspace{1ex}
{\centering Qui visse Torquato Tasso durante il suo
soggiorno del 1605\par}\vspace{1ex}\noindent ma la cosa
gli sembrava strana, poiché ricordava che il
Tasso era vissuto  nel~'500.
```

si ottiene correttamente:

> Andando lungo la strada egli vide una iscrizione che annunciava:
>
> Qui visse Torquato Tasso durante il suo soggiorno del 1605
>
> ma la cosa gli sembrava strana, poiché ricordava che il Tasso era vissuto nel '500.

Nell'esempio precedente, oltre all'illustrazione dell'uso di `\centering` e del comando `\par`, vengono usati diversi altri comandi non ancora descritti, ma che vale la pena di commentare qui.

`\\` serve per andare a capo sia nel mezzo di un testo, sia quando si compongono matrici matematiche o tabelle; questo comando accetta un argomento facoltativo, racchiuso fra parentesi quadre, che indica quanto spazio verticale lasciare dopo essere andati a capo. `\\[1ex]` vuol dire: 'vai a capo e lascia uno spazio verticale pari a 1 ex, cioè pari all'altezza di una lettera 'x' nel font corrente'. Esiste anche la variante `\\*`, che continua ad accettare l'argomento facoltativo, dove l'asterisco impone a TEX di non andare a pagina nuova con la nuova riga.

`\vspace` serve per inserire uno spazio verticale dopo la fine di un capoverso o, detto in termini più tecnici, quando TEX è in modo verticale. L'argomento indica quanto spazio lasciare; siccome questo spazio viene perso all'inizio di una pagina, la forma asteriscata `\vspace*` consente di mantenere lo spazio specificato anche all'inizio della pagina.

`\noindent` serve per evitare il rientro all'inizio di un capoverso.

Attenzione! Sarebbe desiderabile non fare mai uso dei comandi appena descritti; questi non sono parte del mark-up, ma del disegno grafico; sono comandi prescrittivi, non comandi descrittivi. L'autore/compositore dovrebbe attenersi al mark-up descrittivo la maggior parte del tempo. È costretto a ricorrere a questi 'sotterfugi' quando la situazione non prevede qualche ambiente che gli consenta di ottenere la composizione nella maniera desiderata. Piuttosto egli farebbe meglio a definire un nuovo ambiente con le caratteristiche desiderate, ma questo è un argomento che verrà visto molto più avanti.

Nota bene! Al contrario è opportuno usare `\centering` all'interno degli ambienti *figure* e *table* per centrare l'inclusione dell'illustrazione o della tabella da flottare; non occorre servirsi di `\par` perché quegli ambienti contengono già tutte le informazioni per poter gestire questa situazione. Se si usasse invece l'ambiente *center* per centrare l'illustrazione o la tabella, questo ambiente inserirebbe dell'ulteriore spazio bianco prima e dopo la sua apertura e la sua chiusura, per cui l'oggetto flottante risulterebbe poi preceduto e seguito da troppo spazio bianco e/o la didascalia risulterebbe troppo spaziata.

5.8 Le note

LaTeX consente di scrivere due tipi di note, quelle in calce alla pagina e quelle composte nel margine della pagina. Per scrivere note raccolte assieme alla fine di un capitolo, di un articolo, di un libro, bisogna ricorrere al pacchetto esterno **endnote**.

5.8.1 Le note in calce

Per scrivere una nota in calce basta usare il comando `\footnote` con la seguente sintassi:

`\footnote[`⟨*richiamo*⟩`]{`⟨*testo*⟩`}`

Il ⟨*richiamo*⟩ di default è un numero collocato in posizione di apice; talvolta può essere un simbolo come un asterisco, una spada, una doppia spada, eccetera; la numerazione delle note normalmente ricomincia da 1 con l'inizio di un nuovo capitolo, ma la cosa dipende dalla classe del documento. Il ⟨*testo*⟩ della nota non richiede commenti, salvo che si tratta di un 'argomento mobile', e quindi certi comandi 'fragili' non possono essere usati all'interno del ⟨*testo*⟩. Il lettore non si preoccupi dei comandi fragili e degli argomenti mobili; sono cose abbastanza rare quando si scrive un documento 'normale'; in questo testo, che descrive i comandi di LaTeX alcune cose non sono 'normali', proprio perché si tratta di comandi; perciò quando alcuni comandi comparivano nelle note si è dovuto ricorrere a trucchetti vari per poterli scrivere alla lettera, senza farli eseguire.

Le note sono testi speciali, perché vengono composte e trattenute in memoria fino a quando non viene composta la pagina nella quale esse devono comparire; LaTeX fa il possibile affinché le note non siano spezzate fra pagine successive e che ognuna sia collocata al piede della pagina nella quale compare il richiamo. È abbastanza raro che LaTeX non riesca a sistemare la nota in calce senza spezzarla,

ma può succedere. Non è colpa di LaTeX, ma del testo nel quale la nota è richiamata e del testo della nota. Talvolta, cambiando l'uno o l'altro il problema si risolve da solo.

È possibile attribuire una etichetta ad una nota mediante il comando `\label`; e a questa etichetta si può fare riferimento per citare la stessa nota in punti diversi del testo.

All'occorrenza si può fare uso di un comando separato per inserire il richiamo e per scrivere il testo; i comandi seguono la sintassi seguente:

```
\footnotemark[⟨richiamo⟩]
\footnotetext[⟨richiamo⟩]{⟨testo⟩}
```

nei quali il ⟨*richiamo*⟩ è facoltativo; se si fa uso dell'argomento facoltativo sia in questi comandi sia in `\footnote` non viene incrementato il contatore delle note e quindi l'attribuzione di una etichetta mediante `\label` non funziona.

5.8.2 Le note marginali

Se i margini del documento sono abbastanza ampi, questi possono accogliere le note a margine, talvolta chiamate in latino *marginalia*. Il comando per produrle è `\marginpar` e segue la sintassi:

```
\marginapar[⟨nota di sinistra⟩]{⟨nota di destra⟩}
```

Anche le note marginali sono testi che vengono trattenuti in memoria finché la pagina in cui compaiono non viene completamente composta; a seconda del tipo di documento, della classe, dello stile grafico, dell'ampiezza dei margini, la nota marginale può venire collocata nel margine esterno o in quello interno; se però viene collocata a destra del testo, viene composto il testo della ⟨*nota di destra*⟩, altrimenti il testo della ⟨*nota di sinistra*⟩, se è stato specificato, altrimenti il testo della ⟨*nota di destra*⟩ viene usato anche per la ⟨*nota di sinistra*⟩.

Si osservi che se si compone a due colonne, le note relative alla colonna di sinistra vengono collocate a sinistra della colonna e, corrispondentemente le note relative alla colonna di destra vengono collocate a destra della colonna. Componendo ad una colonna le note vengono collocate sempre nel margine esterno, quindi a destra del testo nelle pagine di destra e a sinistra del testo nelle pagine di sinistra. Sembra complicato, ma a pensarci bene è del tutto logico. È vero che con l'apposito comando `\reversemarginpar` questo comportamento può venire scambiato, ma in generale si suppone che questa scelta sia fatta nel file di classe e il compositore non debba preoccuparsene.

Le note marginali vengono composte in colonne strette, con una giustezza che raramente supera i 40 mm; con una giustezza così piccola è possibile che la composizione a pacchetto (giustificata da entrambi i lati) produca una composizione con ampi spazi bianchi fra le parole e con parole difficili da dividere in sillabe che sporgono fuori dalla giustezza. In questi casi sarebbe meglio specificare per la nota di destra una composizione giustificata solo a sinistra; basta usare il comando `\reaggedright`. Per le note di sinistra, per una questione di simmetria, sarebbe quindi desiderabile specificare un testo giustificato solo a destra mediante il comando `\raggedleft`. La cosa è evidentemente possibile se si fa uso dell'argomento facoltativo descritto nella sintassi di `\marginpar`. Tuttavia la lettura di un testo giustificato solo a destra può risultare meno agevole

di un testo giustificato solo a sinistra; ecco perché, pur essendo possibile, raramente si specifica una composizione giustificata solo a destra, anche se i testi delle note marginali sono solitamente molto brevi e quindi non sono faticosi da leggere anche se fossero giustificati solo a destra.

Questa è una nota marginale che si sviluppa su più righe

Questa è una seconda nota marginale abbastanza vicina alla nota precedente

Le note marginali sono collocate nel margine che compete loro mantenendo la loro prima riga allineata con la riga del testo alla quale esse si riferiscono; se due note marginali compaiono troppo ravvicinate e ci fosse il rischio che si sovrappongano, allora LaTeX provvede a spostare in basso le note che potrebbero interferire con le note precedenti assicurando uno spazio di separazione specificato nel file di classe; come si vede nelle due note qui a margine, la seconda, specificata nella riga dove si dice "se due note marginali. . . ", risulta troppo ravvicinata alla prima ed è stata leggermente abbassata.

Le note marginali vengono usate spesso negli scritti letterari, specialmente se si tratta di note brevi e frequenti; è raro incontrare note marginali in scritti tecnici.

Capitolo 6

LaTeX: tabelle

6.1 Introduzione

Le tabelle sono oggetti strutturati, contenenti testo o formule o espressioni matematiche, formati da un certo numero di celle ordinate in senso orizzontale e in senso verticale; queste celle possono essere riquadrate mediante filetti; spesso le colonne di celle hanno ciascuna una prima cella che descrive il contenuto delle celle sottostanti.

Comporre questo insieme di celle è sempre stato un grosso problema anche ai tempi in cui la composizione veniva eseguita a mano; ora i vari word processor e text processor consentono di lavorare con più facilità, ma senz'altro la composizione delle tabelle, specialmente delle tabelle ben composte, richiede molta pazienza e molta preparazione iniziale; se il compositore si fa uno schizzo, anche grossolano, della tabella, sa già in anticipo dove sorgeranno i maggiori problemi e quindi sa predisporre gli accorgimenti adeguati.

Qui si parlerà di tabelle 'testuali'; quelle con contenuto prevalentemente matematico si chiamano in generale *matrici* e verranno viste nel capitolo 10.

Distinguiamo subito due casi, cioè quello delle tabelle che non occupano più di una pagina, e le tabelle lunghe, che occupano più pagine. Le tabelle 'brevi', nonostante la loro brevità, sono oggetti ingombranti, perciò è meglio che il programma di elaborazione del testo, LaTeX nel nostro caso, sposti la tabella nel file composto dove c'è sufficientemente posto per collocarla, talvolta nel punto stesso dove è stata definita, più spesso all'inizio o alla fine della pagina successiva, talvolta alla fine di un capitolo o dell'intero articolo.

Per consentire questo trattamento LaTeX deve conservare in memoria la tabella composta, per poi scaricala nel file di uscita alla prima occasione buona.

Questo modo di procedere implica due cose:

1. le tabelle devono essere numerate, e

2. le tabelle troppo grandi possono bloccare la memoria perché vengono accumulate in memoria in quanto LaTeX non trova mai l'occasione buona per scaricarle nel file di uscita.

Il fatto che le tabelle siano numerate e che debbano essere fatte 'flottare' fino al punto più idoneo richiede un ambiente specifico di 'flottaggio', e implicano

un comando per eseguire la didascalia, anche una semplice didascalia composta dal solo titolino.

6.2 Come far flottare una tabella

L'ambiente di flottaggio si chiama *table* e richiede la seguente sintassi:

```
\begin{table}[⟨codici di posizionamento⟩]
⟨didascalia e tabella vera e propria⟩
\end{table}
```

I ⟨*codici di posizionamento*⟩ sono i codici con i quali viene descritto il modo di collocare la tabella nel testo; di default questi codici sono `t`, `b` e `p`; i codici sono complessivamente i seguenti:

`h`	poni la tabella qui (*here* in inglese)
`t`	poni la tabella in testa alla pagina (*top*)
`b`	poni la tabella al fondo della pagina (*bottom*)
`p`	poni la tabella in una pagina di soli oggetti flottanti (*page*)

6.3 Le didascalie

Quanto segue vale per tutte le didascalie, anche per quelle delle figure. Di default, seguendo la tradizione nordamericana, LaTeX chiede che la didascalia sia posta sotto alle tabelle. Invece per le figure sia la tradizione europea sia quella nordamericana preferiscono la didascalia dopo l'illustrazione. Ovviamente questa è una affermazione generale, visto che le didascalie possono trovare diverse posizioni a seconda del tipo di illustrazioni o di tabelle e del tipo di didascalie.

La didascalia viene composta mediante il comando `\caption` secondo la seguente sintassi:

```
\caption[⟨didascalia breve⟩]{⟨didascalia⟩}
```

La ⟨*didascalia*⟩ è il testo a cui è stato premesso il titolino 'Tabella', o 'Table', o 'Tableau', o ..., a seconda della lingua in uso, seguito dal numero progressivo della tabella; questa parte viene inserita di default; il resto della didascalia può anche essere omessa, ma non sarebbe una buona idea. Il resto della didascalia può essere composto con un titolo seguito da un capoverso descrittivo. Questo capoverso può mancare; se c'è, viene terminato con il punto fermo. Se non c'è, il titolo viene terminato senza punto fermo, come tutti i titoli.

Non è il caso che l'intera ⟨*didascalia*⟩ vada a finire nell'elenco delle tabelle, al massimo ci andrà il titolo, ma non certo il capoverso descrittivo. Ecco, quindi, che la ⟨*didascalia breve*⟩ torna utile per essere inserita nell'elenco delle tabelle, eventualmente abbreviando ulteriormente il titolo inserito nella didascalia completa.

Per modificare lo stile compositivo delle didascalie esiste l'ottimo pacchetto di estensione **ccaption** mediante il quale si possono modificare tutti i possibili parametri che presiedono alla loro composizione.

Descrittore	Spiegazione
`l`	Cella col contenuto allineato a sinistra
`c`	Cella con il contenuto centrato
`r`	Cella con il contenuto allineato a destra
`p{`⟨*larghezza*⟩`}`	Cella contenente un blocco di testo giustificato e largo ⟨*larghezza*⟩
`|`	Filetto verticale
`@{`⟨*espressione*⟩`}`	@-espressione
`\vline`	Filetto verticale da sostituire a `|` dentro una @-espressione
`*{`⟨*numero*⟩`}{`⟨*descrittori*⟩`}`	Ripetizione di ⟨*numero*⟩ volte la stessa sequenza di ⟨*descrittori*⟩

Tabella 6.1: Descrittori delle colonne per le tabelle

6.4 Come comporre la tabella vera e propria

La tabella vera e propria viene composta mediante l'ambiente *tabular* che richiede la seguente sintassi:

```
\begin{tabular}[⟨posizione⟩]{⟨descrittori delle colonne⟩}
⟨prima cella⟩&⟨seconda cella⟩&...\\
⟨prima cella⟩&⟨seconda cella⟩&...\\
...\\
⟨prima cella⟩&⟨seconda cella⟩&...\\
\end{tabular}
```

C'è anche la versione asteriscata che obbedisce alla sintassi seguente:

```
\begin{tabular*}[⟨posizione⟩]{⟨larghezza⟩}{⟨descrittori delle colonne⟩}
⟨prima cella⟩&⟨seconda cella⟩&...\\
⟨prima cella⟩&⟨seconda cella⟩&...\\
...\\
⟨prima cella⟩&⟨seconda cella⟩&...\\
\end{tabular*}
```

6.4.1 I descrittori elle colonne

I descrittori delle colonne sono esposti nella tabella 6.1.

Mentre tutto il resto sembra ovvio, è necessario spiegare che cosa sia una *@-espressione*; questa è una sequenza di comandi, di descrittori, di elementi testuali da mettere fra due celle adiacenti al posto del normale spazio fra le colonne. Anche il descrittore `|` potrebbe essere visto come un separatore fra colonne adiacenti (e lo è) ma non è una @-espressione; questa sostituisce *anche* lo spazio di separazione fra le celle adiacenti, e quindi fra le colonne. Per esempio, si può notare che la tabella 6.1 ha i filetti orizzontali che sporgono un poco fuori dei limiti della cella di sinistra e di destra. Generalmente l'effetto estetico è gradevole, ma potrebbe essere necessario eliminare tale spazio; serve allora inserire come primo descrittore di colonna e come ultimo descrittore di colonna una @-espressione 'vuota': `@{}`.

Esercizio 6.1 Una @-espressione potrebbe essere usata per ripetere come separatore di colonna uno stesso segno; in una tabella che riportasse una colonna con valori numerici fratti e si volesse incolonnare questi numeri in modo che il separatore decimale sia incolonnato fra tutte le celle, indipendentemente dal numero delle cifre prima e dopo la virgola, si potrebbe usare la virgola come testo all'interno della @-espressione spezzando la colonna in due colonne adiacenti e incolonnare la parte intera in una colonna allineata destra e la parte decimale in una colonna allineata a sinistra.

Predisporre una tabella con una colonna di testi descrittivi per ogni riga e una 'colonna' di numeri decimali incolonnata sulle rispettive virgole decimali.

L'esercizio precedente è utile per capire il meccanismo; per fortuna il pacchetto di estensione **array** consente già mediante un nuovo descrittore di incolonnare i numeri fratti sulla base del loro separatore decimale. Le estensioni che questo pacchetto consente sono tali e tante che converrebbe richiamarlo costantemente per ogni documento da comporre. Si rinvia alla documentazione di quel pacchetto per maggiori dettagli.

6.4.2 Il raggruppamento delle celle

Le tabelle possono avere alcune celle adiacenti raggruppate a formare un unica cella. LaTeX di per sé consente di raggruppare le celle solo orizzontalmente; mediante il pacchetto di estensione **multirow** è possibile raggrupparle anche verticalmente, ma il pacchetto è, per così dire, ancora in evoluzione, per cui bisogna supplire a certe piccole manchevolezze con un po' di tentativi e di correzioni successive. In generale, però, una tabella ha un aspetto più professionale se non contiene celle raggruppate verticalmente.

Si usa il comando `\multicolumn` con la sintassi seguente per raggruppare alcune celle orizzontalmente:

`\multicolumn{`⟨*numero*⟩`}{`⟨*descrittore*⟩`}{`⟨*contenuto*⟩`}`

Il ⟨*numero*⟩ specifica quante celle bisogna raggruppare; il ⟨*descrittore*⟩ contiene i codici di allineamento e di separazione necessari per descrivere l'unica grande cella frutto del raggruppamento; il ⟨*contenuto*⟩ è il testo contenuto nell'unica grande cella.

Bisogna fare attenzione ad un dettaglio: i separatori di colonna da inserire nel ⟨*descrittore*⟩ riguardano solo il lato *destro* dell'unica cella, a meno che la cella non sia la prima della serie di celle in una riga, per cui in questo caso e solo in questo caso, bisogna esplicitare anche il descrittore da inserire nel lato sinistro della cella.

Per esempio, se si avesse una tabella in cui il campo dei descrittori fosse il seguente:

```
\begin{tabular}{|*6{c|}}
```

e si volessero raggruppare le celle 3, 4 e 5 in un'unica grande cella presente in una sola riga (per esempio la riga di intestazione), bisognerebbe specificare solamente

```
\multicolumn3{c|}{Intestazione 3}
```

mentre se si volessero raggruppare le celle 1 e 2, sarebbe necessario specificare:

```
\multicolumn2{|c|}{Intestazione 2}
```

Per cui la prima riga della tabella, verosimilmente sarebbe descritta dal seguente codice:

```
\multicolum2{|c|}{Intestazione 2}
             &\multicolum3{c|}{intestazione 3}&Intestazione 1\\
```

Nei precedenti esempi si noti che il numero di celle da raggruppare è formato da una sola cifra quindi non ha bisogno di essere racchiuso fra parentesi graffe; questa è una regola generale; quando un argomento obbligatorio è formato da un solo oggetto (*token* nel gergo di TEX) non ha bisogno delle parentesi graffe, le quali, invece, sono necessarie ogni volta che l'argomento è costituito da diversi token; ovviamente non è vietato usare le graffe anche quando l'argomento è costituito da un solo token, ma omettendole si scrive un po' di meno e non si corre il rischio di dimenticare la graffa chiusa.

6.4.3 I separatori verticali

Per separare verticalmente le righe di celle generalmente non serve nulla; le tabelle sono più professionali se non contengono filetti o se ne contengono pochissimi.

In ogni caso LATEX consente di separare le celle con dei filetti o degli spazi aggiuntivi.

`\\` il comando `\\` serve per terminare una riga di celle; però esso consente anche di specificare uno spazio verticale da aggiungere alla normale spaziatura; si scrive allora, per esempio, `\\[1.5ex]` per terminare la riga e aggiungere uno spazio ulteriore di 1,5 ex fra il fondo della riga corrente e l'inizio della riga seguente. Si faccia attenzione che all'inizio della prima cella di ogni riga di qualunque tabella è inserito un oggetto invisibile che si chiama `\strut` (*pilastrino* in italiano); esso serve per garantire una altezza e una profondità minima ad ogni riga, in modo che le righe sembrino distanziate uniformemente indipendentemente dai loro ascendenti e discendenti; se come argomento facoltativo di `\\` si specifica uno spazio troppo piccolo, questo potrebbe risultare assorbito dalla profondità dello `\strut` della riga corrente.

`\hline` serve per tracciare un filetto orizzontale attraverso tutta la tabella; due comandi `\hline` di seguito l'uno all'altro producono due filetti distanziati di un paio di punti tipografici che rendono maggiormente evidente la separazione verticale. Il comando `\hline` deve essere specificato o prima della prima riga di celle, o dopo un comando `\\`.

`\cline` serve per tracciare un filetto orizzontale sotto *alcune* colonne adiacenti che devono venire specificate mediante la sintassi:

`\cline{`⟨*colonna iniziale*⟩`-`⟨*colonna finale*⟩`}`

Si noti che si possono ripetere due comandi `\cline` di seguito specificando le stesse colonne iniziale e finale in modo da ottenere un filetto doppio, ma non si possono inserire due specificazioni con colonne diverse per mettere due filetti di seguito uno all'altro in orizzontale.

Esercizio 6.2 Si predisponga una tabella a più colonne e si inseriscano contenuti qualsiasi nelle celle; ma si specifichino tutti e tre i tipi di separatori verticali descritti in questo paragrafo, osservando bene che cosa succede quando lo spazio che costituisce l'argomento facoltativo del comando `\\` è di pochi punti. Ripetere la spaziatura con diversi valori di quello spazio.

6.4.4 Come rendere le tabelle un poco più aperte

Nonostante la presenza di uno `\strut` nella prima cella di ogni riga, talvolta sembra che questo `\strut` non sia abbastanza grande cosicché i filetti orizzontali sembrano toccare le lettere maiuscole o i discendenti delle minuscole.

Si può usare il parametro `\arraystretch` con la seguente sintassi:

`\renewcommand\arraystretch{`⟨*fattore*⟩`}`

Il ⟨*fattore*⟩ è un numero decimale, generalmente maggiore di uno, che rappresenta il fattore di scala con cui si vuol ingrandire verticalmente ogni spazio prodotto dagli `\strut`, e quindi con il quale si vuole ingrandire l'intera tabella. Per esempio si può dichiarare:

```
\begin{table}
\renewcommand\arraystretch{1.1}
\begin{tabular}{...}
...
\end{tabular}
\end{table}
```

In questo modo il valore unitario di default del parametro `\arraystretch` viene aumentato del 10% e tutte le righe della tabella risultano del 10% più alte e più profonde.

Attenzione, così facendo si ingrandiscono verticalmente tutte le righe della tabella, anche quelle che non sono precedute o seguite da filetti orizzontali.

Può darsi che questo sia quello che desiderate, specialmente se usate un fattore di scala non tanto superiore a uno. Tuttavia potrebbe essere una scelta non conveniente. Quando volete ingrandire solo le celle precedute e/o seguite da filetti orizzontali non vi resta che inserire voi stessi un altro `\strut` nelle righe che vi interessano. Lo 'strut' che il compositore può usare si chiama `\rule` e la sua sintassi è la seguente:

`\rule[`⟨*innalzamento*⟩`]{`⟨*base*⟩`}{`⟨*altezza*⟩`}`

Questo `\rule` definisce un riga verticale larga ⟨*base*⟩ e alta ⟨*altezza*⟩, innalzata rispetto alla linea di base del testo di ⟨*innalzamento*⟩. Per rendere questa riga invisibile basta specificarne la base nulla; per l'altezza si specifica quello che si vuole e si può innalzare o abbassare (innalzamento negativo) questa riga di quanto si vuole.

Personalmente io uso queste definizioni:

```
\begin{table}
\def\U{\rule{0pt}{3ex}}
\def\D{\rule[-0.5ex]{0pt}{0pt}}
\def\V{\U\D}
\begin{tabular}{...}
...
\end{tabular}
\caption{...}\label{...}
\end{table}
```

e inserisco `\U` nella prima cella di una riga preceduta da un filetto; inserisco `\D` nella prima cella di una riga seguita da un filetto; inserisco `\V` nella prima cella di una riga preceduta e seguita da un filetto.

Il comando `\def` appartiene al linguaggio base di TEX e sarebbe meglio non usarlo mai quando si scrive con LATEX. Infatti questo comando definisce una nuova istruzione e, per sbaglio, si potrebbe ridefinire una istruzione interna di LATEX perché non viene eseguita nessuna verifica della liceità di questa definizione. Tuttavia in questo caso le tre istruzioni `\U`, `\D` e `\V` non corrispondono a nessun'altra definizione interna a LATEX, ma, quello che più conta, la definizione mediante `\def` rimane valida solo all'interno dell'ambiente *table*, per cui all'uscita dall'ambiente, dopo `\end{table}`, quei comandi riprendono il significato che avevano prima di entrare nell'ambiente *table*.

Così è stato fatto in tutte le tabelle esposte finora e così sarà fatto in tutte le prossime tabelle.

6.5 Le tabelle di larghezza specificata

La tabella generata con l'ambiente asteriscato prevede che sia di larghezza specificata proprio mediante l'argomento ⟨*larghezza*⟩.

Per ottenere quella larghezza è necessario che le singole celle possano allargarsi quanto basta perché la tabella complessiva raggiunga il valore desiderato.

A questo scopo LATEX dispone di una specificazione di una lunghezza 'elastica', detta anche 'gomma', che internamente è caratterizzata da tre valori, la lunghezza naturale, l'allungamento e l'accorciamento; se l'allungamento e l'accorciamento corrispondono a valori nulli, essa non si distingue da una lunghezza rigida, che è un altro tipo di lunghezza usato internamente da LATEX.

Se invece l'allungamento e o l'accorciamento corrispondono a valori 'infiniti', si è in presenza di una 'gomma infinitamente allungabile o accorciabile'.

Nel caso delle tabelle si predispone allora una *@-espressione* che al suo interno, oltre agli altri eventuali elementi contenga anche la seguente espressione

```
\extracolsep{\fill}
```

Questa espressione inserisce una lunghezza infinitamente allungabile in tutti i *successivi* separatori di colonna, e questa gomma appare a destra degli altri separatori già presenti; ad esempio:

```
\begin{tabular*}{\textwidth}%
        {@{\extracolsep{\fill}\vline\hspace{\tabcolsep}}l|*3{c|}}
```

permette di definire una tabella la cui prima colonna abbia tutte le celle con il loro contenuto allineato a sinistra e tre altre colonne le cui celle hanno il loro contenuto centrato; le quattro celle di ogni riga, quindi le quattro colonne, sono contornate da filetti verticali; la prima cella, inclusi i separatori, ha le sue dimensioni naturali, mentre le celle successive hanno lo spazio a sinistra di ogni contenuto allargato di quanto basta per allargarle al fine di rendere l'intera tabella larga quanto specificato.

Purtroppo questo modo di procedere va abbastanza bene per tabelle che quasi raggiungono naturalmente la larghezza desiderata, cosicché lo spazio che è necessario aggiungere sia otticamente trascurabile rispetto alla spaziatura normale. Invece se le celle hanno un contenuto abbastanza corto e la tabella deve essere allungata molto, il risultato è tutt'altro che ottimale. Si vedano le tabelle 6.2 e 6.3 per rendersi conto di quanto affermato.

Famiglia			
Rossi Mastrodomenico Papadopoulos	Giovanni Giovanni Battista Filolaos	Maria Maria Concetta Penelopi	Filippo Maria Elena Irini

Tabella 6.2: Tabella di larghezza pari alla giustezza del testo ottenuta allargando poco le singole celle

Cognome	Età	Peso	Altezza
Rossi	50	85	182
Mastrodomenico	65	72	175
Papadopoulos	47	75	160

Tabella 6.3: Tabella di larghezza pari alla giustezza del testo ottenuta allargando troppo le singole celle

Ci sono diverse soluzioni a questo problema; una consiste nel richiamare il pacchetto di estensione **tabularx**, alla cui documentazione si rinvia il lettore.

Un'altra soluzione consiste nel definire un nuovo ambiente che calcoli la larghezza naturale delle singole celle e distribuisca omogeneamente lo spazio mancante a sinistra e a destra di ogni separatore di colonna, in modo che lo spazio bianco necessario per ottenere il risultato desiderato sia distribuito più uniformemente. Nella tabella 6.3 si osserverebbe allora che le colonne 2, 3 e 4, avrebbero davvero il loro materiale centrato, ma anche la colonna 1 avrebbe margini un poco più ampi così da ridurre lo spazio bianco nelle colonne di destra; si osservi il risultato nella tabella 6.4.

In ogni caso ci si ricordi che la presenza dei filetti verticali peggiora solamente l'effetto visivo, se bisogna davvero forzare un poco la composizione; per cui è quanto mai opportuno cercare di evitare l'inserimento dei filetti, in primis quelli verticali, possibilmente anche quelli orizzontali, accettando solo quelli che delimitano la tabella e il filetto sotto alla prima riga che contiene le intestazioni delle colonne. Questa è la raccomandazione stilistica contenuta in moltissimi manuali di tipografia; è giusto che LaTeX consenta di inserire tutti i filetti che

Cognome	Età	Peso	Altezza
Rossi	50	85	182
Mastrodomenico	65	72	175
Papadopoulos	47	75	160

Tabella 6.4: Tabella di larghezza pari alla giustezza del testo ottenuta allargando lo spazio fra le singole celle

si vogliono, ma sta al compositore scegliere se metterli, quali mettere, e dove metterli.

6.6 Problemi compositivi delle tabelle

Le tabelle sono difficili da comporre, non tanto perché i comandi necessari sono complessi, quanto perché esse devono venire progettate con anticipo, non devono essere composte sperando che vada tutto bene.

Ma anche con una buona impostazione il contenuto delle celle richiede aggiustamenti che non possono essere previsti in automatico dal programma, ma devono essere valutati con attenzione dal compositore.

Ecco quindi che nei prossimi paragrafi si daranno indicazioni per rimediare ad alcuni problemi che talvolta si presentano; questi suggerimenti, però, servano anche come stimolo per trovare autonomamente le soluzioni più consone ad altri problemi specifici che potrebbero manifestarsi.

6.6.1 Tabelle troppo larghe

Può darsi che una tabella sia così larga da fuoriuscire dalla giustezza; ma in questo caso si può rimediare in diversi modi a seconda che l'eccesso di larghezza rispetto alla giustezza sia di pochi punti, di pochi millimetri, o invece sia abbastanza consistente.

Nei primi due casi si possono mettere due *@-espressioni* 'vuote' come delimitatori a sinistra della prima cella e a destra dell'ultima cella; così facendo si eliminano le sporgenze dei filetti (se ci sono) fuori della giustezza della tabella.

Se ciò non bastasse, si potrebbe ancora ridurre lo spazio fra le colonne; questo spazio è dato da `\tabcolsep` che di default vale 6 pt. Si noti: questo spazio è quello che viene messo a sinistra e, rispettivamente, a destra di ogni separatore verticale; quindi i contenuti delle celle sono separati dal doppio, vale a dire da 12 pt. Non si perde molto nella composizione della tabella se si imposta il valore di `\tabcolsep` ad un valore di 3 o 4 pt; a seconda del numero delle colonne si possono recuperare diversi millimetri di larghezza.

Un'altra soluzione può consistere nel ridurre il corpo dei font con cui è composta la tabella; passando a `\small` o a `\footnotesize` spesso si risolvono problemi importanti. Basta procedere così:

```
\begin{table}
\small % oppure \footnotesize
\begin{tabular}{...}
...
```

```
\end{tabular}
\caption{...}
\label{tab:...}
\end{table}
```

Se il problema non è molto grave, sarebbe meglio evitare di scendere al corpo corrispondente a `\footnotesize`.

Simile a questa soluzione, esiste la possibilità di ricorrere al pacchetto **graphicx** che fornisce il comando `\resizebox`; questa soluzione è più flessibile, perché consente di scegliere il fattore di scala praticamente con continuità. Il suo unico difetto è che si può usare questa soluzione solo quando il file finale deve essere in formato PostScript oppure PDF, perché solo questi formati sono capaci di scalare con fattori di scala qualunque gli oggetti che contengono. Il file DVI, invece, non è in grado di scalare gli oggetti che contiene.

Resta il terzo caso dove la larghezza della tabella, nonostante le cure descritte nei capoversi precedenti, continua ad eccedere la giustezza del testo; se si tratta di una decina di millimetri al massimo, secondo il giudizio estetico del compositore, la si può collocare centrata lasciandola sporgere metà a sinistra e metà a destra. L'operazione è molto semplice, perché basta racchiudere tutta la tabella dentro una scatola di larghezza specificata; il comando `\makebox` che esegue questa operazione agisce secondo la seguente sintassi:

`\makebox[`⟨*larghezza*⟩`][`⟨*collocazione*⟩`]{`⟨*testo*⟩`}`

dove ⟨*larghezza*⟩ rappresenta la larghezza vera o apparente della scatola così creata; la ⟨*collocazione*⟩ specifica se il `{`⟨*testo*⟩`}` va collocato al centro, oppure allineato a sinistra o a destra dentro la scatola. Il ⟨*testo*⟩ può essere effettivamente del testo comune, ma può anche essere una tabella o qualunque altro oggetto già composto e trattato come un solo oggetto; anche una figura o un diagramma può essere usato come ⟨*testo*⟩. La ⟨*larghezza*⟩ è una larghezza vera se il testo ha una larghezza nominale inferiore quella specificata, mentre essa è una larghezza apparente nel caso opposto; in altre parole se il ⟨*testo*⟩ è più largo di quanto specificato, lo si fa apparire largo esattamente come ⟨*larghezza*⟩.

Per centrare una tabella un pochino più larga della giustezza corrente basta dunque inserirla dentro una scatola larga quanto `\linewidth` specificando il centraggio dentro la scatola (cioè non specificando nessuna ⟨*collocazione*⟩ visto che quella centrata è quella di default). Un costrutto del tipo

```
\begin{table}
\makebox[\linewidth]{%       Inizio scatola
   \begin{tabular}{...}
   ...
   \end{tabular}%
}%                           Fine scatola
\caption{...}\label{tab:...}
\end{table}
```

risolve il problema.

Tuttavia può ancora succedere che la tabella non possa essere 'aggiustata' con nessuno dei trucchi sopra esposti. Allora può succedere che l'eccesso di larghezza rispetto alla giustezza sia ancora dominabile con una rotazione di 90°

in senso *antiorario*. Attenzione: la rotazione deve sempre avvenire in senso antiorario, indipendentemente dal fatto che la tabella venga poi fatta flottare in un recto invece che in un verso. Questo implica che per leggere la tabella, il documento debba venire ruotato sempre in verso *orario* di 90°.

Per eseguire questa rotazione, tenendo conto che la rotazione coinvolge anche la didascalia, è opportuno usare il pacchetto di estensione **rotating** alla cui documentazione si rinvia per la necessaria documentazione.

6.7 Tabelle troppo lunghe

Con 'tabella lunga' si intende una tabella che non stia in una pagina. In questo caso ci sono diverse soluzioni, la prima delle quali è domandarsi se la tabella sia stata concepita bene; potrebbe darsi che scambiando le righe con le colonne la tabella diventi più larga e meno lunga; vale quindi la pena di ripensare alla possibilità di sistemare la tabella con questo semplice artificio.

Se la tabella dovesse risultare troppo larga, ma tale da poter stare in una sola pagina, si possono adottare i rimedi esposti nel paragrafo precedente.

Ma se la tabella è veramente lunga bisogna ricorrere a pacchetti di estensione, i più noti dei quali sono **longtable** e **supertabular** alla cui documentazione si rinvia, visto che questo tipo di composizione richiede un approccio abbastanza delicato.

Concettualmente il problema è semplice; si tratta di spezzare una lunga tabella in monconi tali da poter essere sistemati ciascuno in una pagina sola; per ovvie ragioni estetiche le colonne devono mantenersi della stessa larghezza in tutti i monconi, ma bisogna anche ripetere le intestazioni delle colonne, per non dover sfogliare pagine indietro per sapere se nella terza colonna compare la densità relativa oppure la massa volumica delle sostanze elencate nella tabella.

È opportuno anche che in ogni moncone della tabella che prosegue nella pagina successiva sia inserita una specie di intestazione di piè di pagina che specifica che la tabella continua; analogamente in ogni moncone della tabella, eccetto il primo, conviene mettere una intestazione dalla quale si capisca che si è nel cuore della tabella, non all'inizio né alla fine; questo può ottenersi anche ripetendo la didascalia alla fine della quale sia scritto 'continua', possibilmente fra parentesi e in corsivo, così da separare bene questa informazione dal resto della didascalia. Ovviamente il contatore della tabella non deve aumentare di una unità ad ogni moncone.

Quando si sono ben afferrati questi concetti, per altro abbastanza semplici, allora diventa abbastanza facile comprendere la documentazione dei due pacchetti sopra indicati per potersi regolare al meglio e comporre una tabella professionale.

Un suggerimento non guasta. Conviene inserire il codice della lunga tabella un un file TEX a parte; solo il codice, senza `\documentclass` e l'ambiente *document*. Conviene fare uso del pacchetto **afterpage** e servirsi del comando `\afterpage`. Infatti, benché una lunga tabella non sia flottabile, è opportuno che il primo moncone della tabella cominci all'inizio di una nuova pagina; mediante `\afterpage`, pertanto, diventa quasi spontaneo scegliere una posizione adeguata per dare il comando di leggere e di assorbire, usando il comando `\input`, il file che contiene il codice per la compilazione della tabella; questa automaticamente comincia all'inizio della prima pagina disponibile, finisce qualche

pagina dopo, e il testo ricomincia subito dopo. Come si capisce bene, è estremamente importante scegliere il punto giusto per l'inserimento della lunga tabella; non necessariamente questo punto corrisponde alla fine di un capitolo; dipende dal genere di testo che si sta componendo. Se il codice per la lunga tabella è contenuto nel file `tabellalunga.tex`, memorizzato nella cartella `./file-inclusi`, allora quanto detto sopra corrisponde semplicemente a dare il comando

```
\afterpage{\input{file-inclusi/tabellalunga}}
```

Nell'appendice C si è fatto appunto così.

Si noti ancora che con le tabelle che si sviluppano su più pagine è necessario mettere la didascalia prima della tabella; per questo, a differenza dell'uso americano, rispecchiato in questo libro composto con la classe *book* alla quale non si sono apportate modifiche, sarebbe opportuno mettere sempre la didascalia prima di qualsiasi tabella, non solo prima di quelle lunghe. Per questo scopo, al fine di non dover aggiustare le cose a mano per ogni tabella che si compone, è possibile avvalersi del pacchetto **topcapt** che mette a disposizione del compositore il comando `\topcaption`, da usare esattamente come `\caption`; esso usa le spaziature giuste per mettere la didascalia sopra, invece che sotto, ad un qualunque oggetto flottante; si userà quindi `\topcaption` per le didascalie delle tabelle e `\caption` per le didascalie delle figure.

Nel prossimo paragrafo sarà mostrato un esempio d'uso delle potenzialità dei pacchetti di estensione per le tabelle, descrivendo quanto si è fatto per comporre le lunghe tabelle dell'appendice C.

6.8 Pacchetti di estensione per le tabelle

Oltre ai già citati pacchetti **tabularx**, **longtable** e **supertabular**, conviene riprendere il discorso del pacchetto **array** appena menzionato in relazione all'incolonnamento basato sul separatore decimale. Per dirla tutta, questo lavoro viene eseguito da una piccola estensione del pacchetto **array** che ne sfrutta appieno le potenzialità; si tratta del pacchetto **dcolumn** che è in dotazione standard di ogni distribuzione del sistema TeX.

Questo pacchetto definisce una colonna di nuovo tipo e con il codice `D`; ma accetta diverse personalizzazioni che è meglio esaminare nella documentazione.

Qui si vuole richiamare l'attenzione del lettore sul fatto che il pacchetto **array** mette a disposizione del compositore due nuovi descrittori di colonna simili a `p{...}`; si tratta di `m{...}` e `b{...}`. Agiscono nello stesso modo, nel senso che tutti e tre i descrittori richiedono una larghezza di colonna al posto dei puntini; la differenza è che il contenuto della scatola di testo prodotto con `p{...}` ha la sua prima riga allineata con la prima o unica riga delle celle adiacenti; `b{...}` ha invece la sua ultima riga allineata con le celle adiacenti; infine `m{...}` è centrato con la sua riga mediana rispetto alle celle adiacenti.

Questi tre descrittori sono utilissimi, in particolare lo scrivente ha notato che, secondo i suoi gusti e il tipo di tabelle che compone, le celle con il blocco di testo centrato verticalmente sono più gradevoli rispetto a quando sono allineati sulla base della prima o dell'ultima riga. Ripeto, si tratta di una questione di gusti, ma principalmente del tipo di tabelle che si è soliti comporre, ma è opportuno poter disporre di una scelta.

Il pacchetto **array** consente anche di comporre tabelle miste, in cui alcune colonne sono testuali ed altre puramente numeriche; anzi la caratteristica di **array** è quella di inserire una dichiarazione all'inizio di tutte le celle della stessa colonna e di porre una corrispondente dichiarazione alla fine di ciascuna di queste celle.

I descrittori delle colonne possono diventare particolarmente complessi, ma se si fa attenzione a descrivere con ordine ciascuna colonna, la cosa non è affatto complicata.

I nuovi 'comandi' per inserire queste dichiarazioni iniziali e finali in tutte le celle di una stessa colonna sono `>` e `<`. Essi seguono la sintassi seguente:

```
>{⟨dichiarazioni iniziali⟩}
⟨descrittore di colonna⟩
<{⟨dichiarazioni finali⟩}
```

Val più un esempio per descriverne l'azione, che non tante parole[1]; predisponiamo una piccola tabella con le colonne allineate a sinistra, al centro, e a destra, ma dove la prima colonna deve avere tutti gli elementi in grassetto, la seconda deve contenere una descrizione in corsivo, la terza deve contenere un numero che rappresenta un prezzo e quindi deve essere presente l'unità monetaria; la tabella 6.5 è stata composta con i seguenti comandi

```
\begin{table}\centering
\begin{tabular}{>{\bfseries}l>{\itshape}cr<{\enspace\texteuro}}
\multicolumn3c{\textsf{\bfseries Prezziario}}\\\hline
Pantaloni   & alla zuava     & 35,00 \\
Pantaloni   & corti          & 12,50 \\
Camicie     & a scacchi      & 22,75 \\
Camicie     & botton down    & 31,20 \\
Calze       & di lana        &  8,33 \\
Calze       & di cotone      &  5,00 \\
Calze       & collant        & 15,70
\end{tabular}
\caption{...}\label{tab:arraypack}
\end{table}
```

e si vede quanto sia conveniente l'uso dei descrittori iniziali e di quelli finali per la composizione di colonne omogenee.

Ma una delle cose più utili del pacchetto **array** è costituito dalla possibilità di definire nuovi descrittori di colonna che possono ulteriormente semplificare la composizione delle tabelle.

Infatti il pacchetto **array** mette a disposizione il comando `\newcolumntype` con la seguente sintassi:

```
\newcolumntype{⟨descrittore⟩}{⟨specifiche⟩}
```

Il ⟨*descrittore*⟩ è formato da una sola lettera minuscola o maiuscola che non sia già stata usata per i descrittori esistenti e le ⟨*specifiche*⟩ sono costituite

[1] In questo stesso paragrafo si vedrà più avanti come è stata composta la tabella C.8 facendo uso di questi descrittori speciali.

Prezziario		
Pantaloni	*alla zuava*	35,00 €
Pantaloni	*corti*	12,50 €
Camicie	*a scacchi*	22,75 €
Camicie	*botton down*	31,20 €
Calze	*di lana*	8,33 €
Calze	*di cotone*	5,00 €
Calze	*collant*	15,70 €

Tabella 6.5: Tabella composta con le estensioni del pacchetto **array**

da un descrittore completo delle sue specificazioni particolari descritte mediante i comandi `>{...}` e/o `<{...}`; estendendo il concetto, il ⟨*descrittore*⟩ può corrispondere a delle ⟨*specifiche*⟩ che fanno uso di più descrittori.

Per esempio, nella tabella 6.5 sono stati usati per le tre colonne i descrittori

```
>{\bfseries}l>{\itshape}cr<{\enspace\texteuro}
```

Facendo uso del comando appena descritto si sarebbero potuti definire tre nuovi descrittori:

```
\newcolumntype{L}{>{\bfseries}l}
\newcolumntype{C}{>{\itshape}c}
\newcolumntype{E}{r<{\enspace\texteuro}}
```

e poi si sarebbe potuto descrivere il comando di apertura della tabella con:

```
\begin{tabular}{LCE}
```

È chiaro che se questi tipi di colonne vengono usati spesso, è conveniente eseguire le definizioni dei nuovi tipi di colonne una volta per tutte nel preambolo e poi usarli regolarmente nelle varie tabelle che si intendono comporre.

Al limite, se diverse tabelle usano la stessa sequenza di descrittori di colonna, questa può essere inclusa in un unico descrittore; per esempio

```
\newcolumntype{T}{LCE}
...
\begin{tabular}{T}
```

Anche nelle lunghe tabelle dell'appendice C si è fatto nello stesso modo.

Per comodità del lettore, si riporta qui l'insieme delle dichiarazioni usate per comporre la tabella C.8, compreso l'inizio e la fine della tabella, ma senza il contenuto, per ovvi motivi di brevità.

```
\newcommand*{\hstrut}{\rule{0pt}{3ex}\hskip 0sp}
\newcolumntype{B}{>{\bfseries}c}
\newcolumntype{M}{>{$\displaystyle}c<{$}}
\newcolumntype{R}{>{\raggedright}p{.35\textwidth}}
\newcolumntype{S}{>{\hstrut}p{.45\textwidth}<{\vspace*{1ex}}}
...
\tablecaption{Simboli matematici}\label{tmat}
\tablefirsthead{%
     \noalign{\hrule\medskip}%
```

```
     \multicolumn1B{Simbolo}&
     \multicolumn1B{Significato}&
     \multicolumn1B{Note}\\
     \noalign{\smallskip\hrule\medskip}}
\tablehead{\multicolumn3l{\emph{Continua tabella \thetable}}\\
     \noalign{\smallskip\hrule\medskip}
     \multicolumn1B{Simbolo}&
     \multicolumn1B{Significato}&
     \multicolumn1B{Note}\\
     \noalign{\smallskip\hrule\medskip}}
\tabletail{\noalign{\smallskip\hrule\smallskip}%
     \multicolumn3r{\emph{continua}}\\}
\tablelasttail{\noalign{\smallskip\hrule}}
%
\begin{supertabular*}{\textwidth}{@{\extracolsep{\fill}}MRS@{}}
... & ... & ... \\
\end{supertabular}
```

Come si vede `\tablecaption` permette di comporre la didascalia della lunga tabella al punto giusto, prima della tabella. `\tablefirsthead` consente di comporre le intestazioni delle colonne nel primo moncone. `\tablehead` consente di comporre l'intestazione delle colonne nei monconi successivi al primo; nello stesso tempo esso contiene anche l'informazione che la tabella che compare nella pagina corrente è una cotinuazione di una tabella iniziata in pagine precedenti e il riferimento `\thetable` serve appunto per ripetere il numero della tabella senza incrementarne il contatore. `\tabletail` consente di comporre la fine di ogni moncone tranne l'ultimo. `\tablelasttail` consente di comporre la fine dell'ultimo moncone. I comandi per queste composizioni vengono memorizzati ed eseguiti ogni volta che ce ne sia la necessità, perché l'ambiente *supertabular* provvede da solo a spezzare la tabella dove è necessario.

Fra i comandi usati si notano alcuni comandi primitivi, come `\noalign` che serve per inserire qualcosa dentro ad un tabella, senza tenere conto dei descrittori delle colonne; `\hrule` indica un filetto orizzontale che attraversa tutta la tabella; `\medskip` indica di inserire uno spazio verticale medio (circa mezza riga); `\smallskip` indica di inserire uno spazio verticale piccolo (circa di un terzo di riga). `\emph` serve per evidenziare una parola o una breve locuzione, nel senso che, se essa va evidenziata all'interno di un brano composto in tondo, esso evidenzia il suo argomento componendolo in corsivo; al contrario esso evidenzia in tondo all'interno di un brano in corsivo. La prima colonna è composta in modo matematico e in stile `\displaystyle`; la seconda colonna è composta in bandiera allineata a sinistra con una giustezza di 0,35 volte la giustezza della pagina; la terza colonna è composta con una giustezza pari a 0,45 volte la giustezza della pagina, ma dopo aver inserito uno strut orizzontale definito come uno strut alto 3 ex seguito da uno spazio nullo[2]; questo spazio nullo rappresenta un trucchetto per assicurare che quanto segue sia eventualmente divisibile in sillabe (vedi appendice D). Il comando `\thetable` serve per scrivere il contenuto del contatore `table`.

[2] L'unità di misura `sp` indica la frazione $1/2^{16}$ di un punto tipografico; la notazione in base 2 è chiarissima per un calcolatore, mentre la notazione decimale 1/65536 non è altrettanto significativa.

Nel paragrafo 16.9 sono esposti alcuni altri argomenti particolari relativi alle tabelle e alle figure inserite in alcune celle. Vi si definiscono dei nuovi comandi, ed è per questo che l'argomento è rinviato a quel paragrafo.

Capitolo 7

LaTeX: figure

7.1 Le figure e le immagini

Con LaTeX è opportuno distinguere bene fra 'figura' e 'immagine' o 'disegno' o 'grafico'; la *figura* è un oggetto non verbale (fotografia, disegno, grafico, schema, diagramma, eccetera) dotato di un numero e di una didascalia che LaTeX deve sistemare in qualche posto dove ci sia abbastanza spazio, visto che generalmente l'oggetto è di dimensioni relativamente grandi.

Invece l'*oggetto grafico* che viene trattato da LaTeX come una figura può avere le forme più disparate e può avere anche le origini più diverse. Potrebbe essere prodotto con una macchina fotografica, con un software da disegno, con i comandi stessi di LaTeX e dei suoi pacchetti di estensione; potrebbe essere costituito da codice di programmazione che verrà tradotto in grafico dal driver che rende leggibile agli umani il documento.

In questo capitolo si parlerà solo dell'ambiente *figure* e dei modi di produrre materiale grafico con LaTeX stesso, accennando appena ai disegni eseguibili con i pacchetti di estensione.

7.2 L'ambiente `figure`

L'ambiente *figure* è del tutto analogo all'ambiente *table*; serve per rendere flottante un oggetto e per dargli una didascalia il cui primo termine è il nome 'Figura, o 'Figure, o 'Abbildung' o ..., a seconda della lingua in uso, seguito dal suo numero progressivo e dalla didascalia vera e propria, composta come sempre da un titolo e da una spiegazione (facoltativa).

La sintassi è simile a quella dell'ambiente *table*:

```
\begin{figure}[⟨posizione⟩]
...
\end{figure}
```

e i parametri di ⟨*posizione*⟩ `h`, `t`, `b` e `p` sono gli stessi dell'ambiente *table*.

Fin qui la differenza rispetto all'ambiente *table* consiste solo nel fatto che viene scritto nel titolino la parola 'Figura' invece della parola 'Tabella'. In realtà il contatore delle figure è diverso da quello delle tabelle, così come la

coda delle figure, o meglio, la gestione della coda delle figure è diversa e distinta da quella delle tabelle. Ci si ricordi solo che oggetti troppo grandi, siano essi figure o tabelle, possono bloccare il deflusso degli oggetti della stessa specie dalla rispettiva coda e questo potrebbe portare non solo allo 'scarico' di tutti questi oggetti alla fine del capitolo o del documento, ma potrebbe portare anche alla saturazione delle code e quindi ad una situazione di stallo nella esecuzione della compilazione del documento.

Un modo di evitare questo stallo è quello di usare il pacchetto esterno **afterpage** il quale mette a disposizione del compositore il comando `\afterpage` con la seguente sintassi

```
\afterpage{⟨azione⟩}
```

dove ⟨*azione*⟩ rappresenta il comando da eseguire alla fine della pagina corrente, quando, cioè, la pagina corrente viene accodata al file di uscita. In quel momento, subito dopo l'accodamento, LaTeX può eseguire l'⟨*azione*⟩; nella fattispecie l'⟨*azione*⟩ prescritta potrebbe essere `\clearpage` che, oltre a terminare la pagina corrente (già terminata) serve anche per scaricare le code. Se si specifica spesso

```
\afterpage{\clearpage}
```

si ottiene il risultato di non accumulare troppo materiale nelle code delle figure e delle tabelle; magari durante la revisione delle bozze si può decidere se e come modificare figure o tabelle per rendere il deflusso dalle code un poco più ordinato in modo che non sia necessario ricorrere ad `\afterpage`.

7.3 L'ambiente `picture`

LaTeX dispone di un meccanismo per produrre disegni al tratto, semplici schemi a blocchi, semplici diagrammi e simili disegnini mediante l'ambiente *picture*.

L'ambiente *picture* sta a *figure* come *tabular* sta a *table*; forse questo paragone serve a comprendere meglio la differenza fra figura e immagine.

L'ambiente *originale*, nato insieme al vecchio LaTeX 209, era molto limitato nella scelta delle linee che potevano essere tracciate, perché venivano tracciate usando dei segmentini tratti da una polizza di font specialmente predisposti per questo scopo; per quanto potessero essere ben disegnati, questi segmentini erano pur sempre in numero limitato.

Dal 2003 esiste l'estensione standard **pict2e** di LaTeX che non è ancora direttamente incorporata nel nucleo di LaTeX, ma che dovrebbe diventarlo in non molto tempo; questa estensione è progettata per eliminare tutte le limitazioni del vecchio ambiente e per aggiungere prestazioni nuove; pertanto si invocherà questa estensione mettendo nel preambolo il codice seguente:

```
\usepackage[⟨opzioni⟩]{pict2e}[2004/08/01]
```

dove ⟨*opzioni*⟩ può essere una o più fra le opzioni disponibili, la maggior parte delle quali servono per definire il tipo di driver che verrà usato per tradurre il risultato in una forma comprensibile agli umani. Per lo più servono solo in casi particolarissimi e rari, ma ci sono altre tre opzioni interessanti:

original serve a impostare **pict2e** in modo che sia perfettamente equivalente al vecchio LaTeX; evidentemente serve solo per compatibilità con il passato.

ltxarrows serve per specificare a **pict2e** che le punte delle frecce devono essere disegnate come vengono disegnate da LaTeX in modo di compatibilità con il passato; siccome quelle frecce con punta triangolare leggermente concave sono molto gradevoli, il compositore può decidere di usare tali punte di freccia, anche se non introducono nessun elemento di novità nello stile del disegno.

pstarrows serve invece a disegnare le punte delle frecce come un aereo stealth, cioè come un aereo con le ali a V; questa è la maniera di disegnare le frecce usata anche dal pacchetto PSTricks, eccellente pacchetto di estensione che si affida completamente al driver dvips per produrre un file di tipo PostScript.

Se non si specificano opzioni, il pacchetto cerca se esiste il file di configurazione `pict2e.cfg` e sceglie le opzioni che quel file riesce a definire in base al programma eseguito; inoltre sceglie di default le frecce tradizionali di LaTeX.

A parte questi preliminari, l'ambiente *picture* definisce uno spazio grafico in cui l'unità di misura è definita dal parametro `\unitlength`, che di default vale 1 pt. Specificando un valore diverso è facilissimo scalare il disegno in modo da ingrandirlo o da rimpicciolirlo. Come trucco personale io preferisco definire per ogni ambiente *figure*, prima di aprire l'ambiente *picture*, una unità di misura dipendente dal font in uso, per esempio 1 ex, così che cambiando il corpo del font anche il disegno viene scalato proporzionalmente. In altre circostanze scelgo come unità di misura un centesimo della giustezza; in questo modo sono sicuro che, se non supero il valore di 100 unità in orizzontale, il disegno certamente non straborda fuori dei margini. La sintassi da usare per impostare l'unità di misura è:

```
\setlength{\unitlength}{⟨lunghezza⟩}
```

Dopo questa prima operazione l'ambiente *picture* con le sue dimensioni apparenti e il suo offset, nonché ogni distanza o lunghezza a cui si faccia riferimento all'interno di questo ambiente, è indicata mediante un numero, senza specificare le unità di misura, perché queste, è sottinteso, sono identificate da `\unitlength`.

La sintassi di apertura dell'ambiente *picture* è la seguente:

```
\begin{picture}(⟨base⟩,⟨altezza⟩)(⟨x-offset⟩,⟨y-offset⟩)
...
\end{picture}
```

Si notino le parentesi tonde per specificare i multipli dell'unità di misura che si intendono usare; la prima coppia di parentesi tonde contiene le dimensioni della base e dell'altezza 'apparenti' del disegno; *apparenti* significa che il disegno verrà inserito dentro un rettangolo di quelle dimensioni, ma alcune parti del disegno potrebbero fuoriuscire da quel rettangolo, al punto che sarebbe (e di fatto è) possibile dichiarare entrambe le dimensioni pari a zero. La seconda coppia di parentesi è facoltativa; se c'è, essa indica la posizione dello spigolo inferiore sinistro del suddetto rettangolo rispetto all'origine degli assi x e y; questi sono gli

assi rispetto ai quali vengono specificate le coordinate degli oggetti da collocare nel disegno.

Ogni oggetto da collocare nel disegno viene collocato mediante il comando `\put`; se l'oggetto deve essere collocato un certo numero di volte in posizioni regolarmente spaziate si può usare il comando `\multiput`; le sintassi sono le seguenti:

```
\put(⟨x⟩,⟨y⟩){⟨oggetto⟩}
\multiput(⟨x-ini⟩,⟨y-ini⟩)(⟨x-step⟩,⟨y-step⟩){⟨numero⟩}{⟨oggetto⟩}
```

Le coordinate ⟨*x-ini*⟩ e ⟨*y-ini*⟩ sono le coordinate del primo punto; le coordinate ⟨*x-step*⟩ e ⟨*y-step*⟩ sono gli incrementi da dare ad x e ad y ogni volta che si deve inserire una nuova istanza dell'oggetto; il ⟨*numero*⟩ indica il numero totale di istanze dell'oggetto; infine ⟨*oggetto*⟩ è l'oggetto grafico da collocare nel disegno.

Gli oggetti collocabili sono linee, vettori (frecce), cerchi, dischi, testi semplici o riquadrati con linee continue o tratteggiate; i testi possono essere collocati intelligentemente rispetto al rettangolo ideale che li contiene. Possono poi essere collocati degli 'ovali', cioè dei rettangoli con gli spigoli arrotondati, dentro ai quali può essere collocato del testo; possono essere fatti semplici disegni tracciando delle curve mediante l'uso di curve di Bézier di secondo o di terzo grado. La sintassi che descrive ognuno di questi oggetti è la seguente:

```
\line(⟨x-pend⟩,⟨y-pend⟩){⟨lunghezza⟩}
\vector(⟨x-pend⟩,⟨y-pend⟩){⟨lunghezza⟩}
\circle{⟨diametro⟩}
\circle*{⟨diametro⟩}
\makebox(⟨base⟩,⟨altezza⟩)[⟨posizione⟩]{⟨testo⟩}
\framebox(⟨base⟩,⟨altezza⟩)[⟨posizione⟩]{⟨testo⟩}
\dashbox{⟨lung-trattino⟩}(⟨base⟩,⟨altezza⟩)[⟨posizione⟩]{⟨testo⟩}
\oval[⟨raggio⟩](⟨base⟩,⟨altezza⟩)[⟨parte⟩]
```

I nomi dei parametri dovrebbero essere abbastanza chiari, ma vanno specificate alcune cose particolari.

La lunghezza di segmenti e vettori è la componente orizzontale, cioè la proiezione sull'asse x del segmento o del vettore; se questo è completamente verticale allora e solo allora la lunghezza coincide con quella del segmento o del vettore.

I coefficienti di pendenza dei segmenti e dei vettori rappresentano le proiezioni lungo l'asse x e y rispettivamente di un tratto di segmento o di vettore; siccome queste proiezioni non possono essere numeri arbitrariamente grandi e siccome devono essere numeri interi il cui valore assoluto non superi 999, è evidente che conviene scegliere i valori più piccoli conformi a queste limitazioni arrotondando ai valori interi più vicini. Per un vettore verticale che ha la punta della freccia in alto questi coefficienti formeranno la coppia `(0,1)`, ma si potrebbe indicare anche `(0,999)` con il medesimo risultato; per ovvi motivi sarebbe meglio usare la prima scrittura. Per i segmenti, che non hanno una punta di freccia che ne indichi la direzione, questa va intesa come la direzione del segmento dal suo estremo collocato con `\put` all'altro estremo; in altre parole il comando `\put` colloca alla sua coordinata il primo punto del segmento, da cui parte il segmento con la pendenza specificata.

Il comando `\circle` specifica una circonferenza completa di cui si specifica il ⟨*diametro*⟩; il suo punto di riferimento messo in posizione con `\put`, è il suo centro. Il comando `\circle*` specifica invece un disco completo, non solo il contorno, quindi disegna un cerchio 'nero' con il ⟨*diametro*⟩ specificato.

I comandi `\makebox`, `\framebox` e `\dashbox` definiscono tre scatole, la prima senza che ne venga disegnato il contorno, la seconda con il contorno disegnato, la terza con il contorno tratteggiato con trattini lunghi ⟨*lung-trattino*⟩. Il comando `\put` mette in posizione il loro punto di riferimento, cioè lo spigolo inferiore sinistro. All'interno di questi rettangoli, o scatole, può essere posto del testo che viene collocato rispetto ai bordi ideali, disegnati o tratteggiati conformemente ai parametri di ⟨*posizione*⟩; questi sono le solite lettere `t`, `b`. `l`, `r` e `c`, con il rispettivo significato di 'top', 'bottom', 'left', 'right' e 'center'; per ogni oggetto si possono specificare un solo parametro o due parametri di posizione, ricordando che 'center' è sempre il valore di default.

È particolarmente comodo il comando `\makebox` con dimensioni di ⟨*base*⟩ e ⟨*altezza*⟩ nulle, perché il ⟨*testo*⟩ che vi viene inserito, risulta collocato con precisione rispetto allo spigolo inferiore sinistro, ma senza spazio alcuno interposto; non ha importanza che le dimensioni apparenti della scatola siano nulle; è importante che il testo sia collocato con un riferimento preciso, indipendentemente dalla presenza di ascendenti o discendenti.

Per il comando `\oval` è possibile specificare il raggio minimo del quarto di cerchio che costituisce lo spigolo; per altro le dimensioni di ⟨*base*⟩ e ⟨*altezza*⟩ di questo speciale rettangolo a spigoli arrotondati si riferiscono al rettangolo completo; ma la specifica facoltativa di ⟨*parte*⟩ permette di scegliere quali angoli mostrare o quale coppia di angoli adiacenti mostrare, così che specificando raggio, metà della base e metà dell'altezza uguali è possibile disegnare solo un quarto di circonferenza oppure metà circonferenza; al solito la ⟨*parte*⟩ è specificata con le lettere `l`, `t`, `b` e `r` (`c` non avrebbe senso), cosicché `t` permette di disegnare solo la metà di sopra, `tl` solo il quarto in alto a sinistra.

Nell'ambito dell'ambiente *picture* è possibile specificare due spessori predefiniti per le linee da tracciare, `\thinlines` e `\thicklines`; si può specificare un valore assoluto di spessore (espresso con le unità di misura) mediante

```
\linethickness{⟨spessore assoluto⟩}
```

Infine i comandi `\qbezier` e `\cbezier` consentono di mettere in posizione delle curve di secondo o di terzo grado rispettivamente, senza bisogno di ricorrere ai comandi `\put` o `\multiput`. La loro sintassi è la seguente:

```
\qbezier(⟨x1⟩,⟨y1⟩)(⟨x2⟩,⟨y2⟩)(⟨x3⟩,⟨y3⟩)
\cbezier(⟨x1⟩,⟨y1⟩)(⟨x2⟩,⟨y2⟩)(⟨x3⟩,⟨y3⟩)(⟨x4⟩,⟨y4⟩)
```

Per la curva di secondo grado, descritta da `\qbezier`, (x_1, y_1) rappresentano le coordinate del punto da cui la curva parte in direzione del punto avente le coordinate (x_2, y_2); invece (x_3, y_3) rappresentano le coordinate del punto di arrivo con la direzione che proviene da (x_2, y_2).

Per la curva di terzo grado, descritta da `\cbezier`, questa parte da (x_1, y_1) in direzione (x_2, y_2) e arriva al punto (x_4, y_4) con la direzione proveniente da (x_3, y_3).

Scegliendo accuratamente i punti iniziali e finali, nonché le direzioni delle tangenti si possono facilmente disegnare semplici diagrammi come mostrato nelle

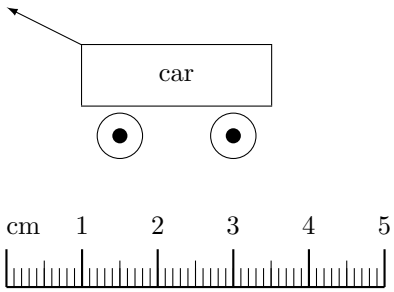

Figura 7.1: Il semplice disegno usato da Leslie Lamport per descrivere le potenzialità dell'ambiente *picture*

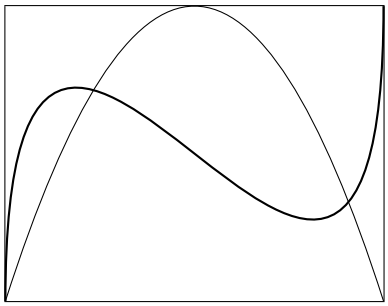

Figura 7.2: Alcune curve di Bézier di secondo e di terzo grado tracciate nell'ambiente *picture*

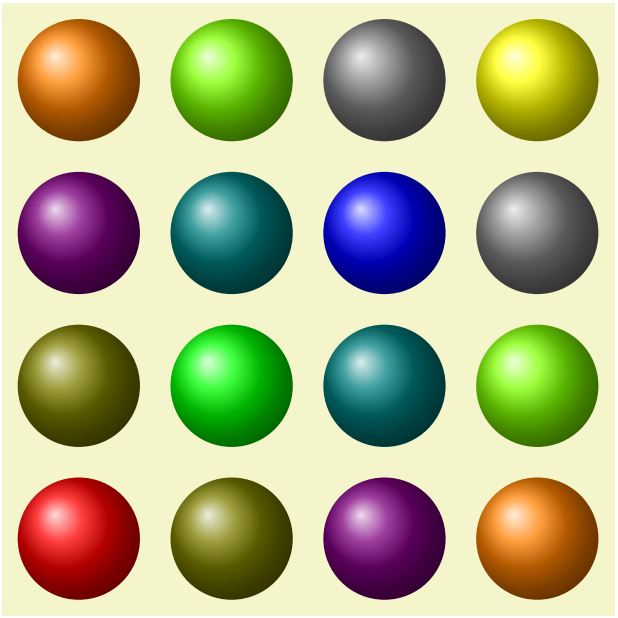

Figura 7.3: Un disegno a colori ottenuto con l'uso del pacchetto **pgf**

figure 7.1 e 7.2. I punti guida delle curve di Bézier non compaiono perché cadono fuori dal rettangolo della figura.

7.4 Il pacchetto pgf

Il pacchetto di estensione **pgf** è ancora in evoluzione ma certamente le potenzialità che ha oggi verranno sicuramente mantenute ad ampliate in futuro. L'autore cerca di costruire una interfaccia utente che prescinda completamente dal particolare strumento informatico che trasforma il codice in qualcosa comprensibile dagli umani. Oggi si possono ottenere risultati ottimi con il pacchetto di

estensione **PSTricks**, ma il suo codice è interpretabile solo dal driver dvips; per terminare con un file in formato PDF è necessario procedere ad una ulteriore conversione di formato[1].

L'acronimo PGF indica 'Portable Graphics Format' e vorrebbe poter un giorno essere altrettanto potente di **PSTricks** ma svincolato da questo o quel driver; nel momento in cui il sorgente `.tex` viene compilato, il motore di compilazione determina con quale programma questa compilazione viene eseguita e sceglie l'interfaccia giusta in relazione a quella situazione; se il file sorgente, come succede spesso, viene spedito per posta elettronica ad un conoscente che lavora con una piattaforma diversa e una distribuzione del sistema TEX diversa o è abituato ad usare un compilatore diverso, ebbene la grafica deve risultare del tutto identica a come riusciva all'autore originale.

Qui sarebbe fuori luogo scendere nei dettagli, e si rinvia il lettore alla documentazione del pacchetto **pgf**; sappia comunque il lettore che questo pacchetto mette a disposizione un ambiente di disegno che si chiama *tikzpicture*, all'interno del quale la sintassi dei comandi è molto diversa dalla sintassi da usare nell'ambiente *picture*, ma sotto certi aspetti essa è più intuitiva, in ogni caso più adatta alla moltitudine di oggetti, linee, colori, nodi, alberi, strutture, schemi di flusso, eccetera, che possono essere gestiti con quel pacchetto e quell'ambiente.

Qui, giusto per far venire l'acquolina in bocca con un blando aperitivo, si mostra solo la figura 7.3 che mostra una figura a colori, dove i colori delle sfere che giacciono sulla diagonale secondaria sono mescolati fra di loro nelle posizioni fuori da quella diagonale, mostrando sempre delle mezze sfere con gradienti di luce obliqua a simmetria circolare.

7.5 I vantaggi di usare programmi nativi del sistema TEX

I vantaggi di usare i programmi nativi del sistema TEX (**pgf** funziona anche con plain TEX e con ConTEXt) consistono nel fatto che questi sistemi consentono di usare direttamente gli stessi font che vengono usati per il testo. Non si trascuri questa possibilità, perché l'uso di font diversi, talvolta incompatibili o non adatti a quello che si sta scrivendo, è un difetto così grave che salta all'occhio anche della persona più inesperta.

I disegni, per esempio, prodotti con famosi programmi interattivi di matematica, possono essere salvati in formato PostScript; tuttavia è poi necessario editare questi file per poter cambiare i font, specialmente quando alcune scritte che compaiono nei disegni sono costituite da formule e chiunque vede subito che non sono composte con la stessa maestria di quelle composte con LATEX. Esistono dei pacchetti, per esempio **psfrag**, che riescono ad eseguire alcune di queste trasformazioni, ma resta sempre un lavoro maggiore da fare che richiede una attenzione maggiore, come succede sempre quando si eseguono delle correzioni.

[1] Questa limitazione, che provoca una apparente incompatibilità fra il compilatore LATEX e il compilatore pdfLATEX, può essere aggirata attraverso il pacchetto **pst-pdf**, alla cui documentazione si rinvia il lettore; un altro pacchetto **auto-pst-pdf** offre una comoda interfaccia per l'uso di **pst-pdf** così che di fatto, sia pure con un lieve rallentamento della compilazione, anche il compilatore pdfLATEX riesce a gestire file sorgente che fanno uso di **PSTricks**. Il lettore interessato può esplorare CTAN`/macros/latex/contrib/pst-pdf/` e CTAN`/macros/latex/contrib/auto-pst-pdf/`.

Non si trascuri, quindi, la possibilità di disegnare semplici diagrammi ricorrendo anche al più elementare ambiente da disegno costituito da *picture*. Il vantaggio dell'uso dei font giusti supera qualunque fatica si debba fare per ottenere un risultato grafico accettabile. Se poi ci si prende il disturbo di imparare ad usare l'ambiente *tikzpicture* allora non si deve rinunciare proprio a niente.

Capitolo 8

LaTeX: l'importazione di figure esterne

8.1 Introduzione

Come accennato nel capitolo precedente, capita di dover importare disegni o immagini prodotte con strumenti esterni. Per quel che riguarda i disegni al tratto si è già spiegato quali vantaggi ci siano ad usare i programmi da disegno nativi del sistema TeX. Tuttavia può succedere di dover importare disegni o immagini prodotti all'esterno.

Qui ci concentreremo sulle immagini, sostanzialmente le fotografie o altri disegni a mezze tinte non ottenibili da un semplice programma specializzato, per esempio, nel tracciare diagrammi a due o a tre dimensioni.

Il pacchetto **pgf** gestisce anche l'importazione delle immagini esterne, ma lo strumento principe per operare in questa direzione è senz'altro il pacchetto **graphicx** che da anni costituisce l'interfaccia di riferimento.

Prima però di addentrarci nell'uso di quel pacchetto bisogna fermarsi un attimo per meditare sulla moltitudine di formati grafici che rendono le cose più difficili di quanto potrebbero essere.

8.2 I formati grafici

I formati grafici si distinguono sostanzialmente in formati vettoriali e in formati a matrici di pixel. Essi hanno caratteristiche diverse e per certe applicazioni vanno meglio i primi, mentre per certe altre vanno meglio i secondi.

8.2.1 I formati vettoriali

I formati vettoriali sono descritti sostanzialmente mediante una specie di programma che contiene le istruzioni per tracciare i contorni delle aree dove il colore è sostanzialmente costante; questi contorni sono costituiti dalla conoscenza delle serie di punti per i quali devono passare le curve di Bézier che li descrivono. Questi formati sono usatissimi per tracciare quei particolari disegni costituiti dai caratteri dei vari font; questi infatti devono potersi disegnare rapidamente, ma, specialmente se devono essere presentati sullo schermo, devono essere scalabili a

piacere senza che l'ingrandimento o il rimpicciolimento producano difetti degni di nota.

È chiaro che se il disegno è costituito dai contorni delle aree delle stesso colore, il problema del cambiamento di scala ha una soluzione banale, in quanto basta moltiplicare le coordinate dei punti che descrivono il contorno per il fattore di scala; per il resto non cambia praticamente nulla. In realtà, quando si rimpicciolisce un disegno che contiene delle linee sottili, queste diventano ancora più sottili e, per esempio, sullo schermo, che ha una densità di puntini luminosi abbastanza bassa, da 70 a 100 puntini al pollice, queste linee diventerebbero troppo sottili per essere rappresentate; l'algoritmo di rappresentazione deve contenere quindi dei suggerimenti (*hints*) per produrre un effetto finale che non lasci sparire le linee troppo sottili.

In questo testo l'argomento diventerebbe troppo tecnico e non si insiste oltre; tuttavia vale la pena di ricordare che i formati grafici vettoriali più noti e frequenti sono il formato PostScript (estensione del file `.ps`) e il suo parente stretto Encapsulated PostScript (estensione `.eps`); oggi sta diventando sempre più frequente il formato Scalable Vector Graphics (estensione `.svg`), usato specialmente per le applicazioni Web.

C'è anche il formato di uscita del programma METAPOST che, partendo da un file sorgente simile a quanto si potrebbe scrivere con il programma di creazione dei font creato da Donald E. Knuth, produce un'uscita in una specie di linguaggio PostScript ridotto e semplificato. Il programma è molto utile per produrre disegni al tratto, che vanno dai disegni ai diagrammi, ma l'unico difetto è che il linguaggio non è dei più semplici e quindi il programma è poco usato, comunque meno di quanto meriterebbe.

A stretto rigore il formato Portable Document Format (estensione `.pdf`) sarebbe un formato vettoriale e sotto molti aspetti lo è. Tuttavia il file è un file con il codice compresso (zippato, secondo una diffusa terminologia), ma il file decompresso è sostanzialmente un sotto insieme del formato PostScript, quindi sostanzialmente vettoriale. Il suo tipo di compressione richiede programmi speciali per la visualizzazione e stampa, destinati specialmente a rendere la pagina sotto forma grafica.

8.2.2 I formati diversi da quelli vettoriali

Le fotografie sono, oggi più spesso di ieri, in formato JPEG (estensione `.jpg`). Tuttavia i formati di matrici di pixel sono diffusissimi, e le estensioni vanno da `.bmp` a `.wmf`, da `.tiff` a `.gif` e a Portable Network Graphics (estensione `.png`); questa è solo una piccola elencazione di formati grafici, perché esistono dei formati specifici per certi particolari apparecchi fotografici digitali o per certi codici di colore.

Il vantaggio dei formati a matrici di pixel è che i programmi per la loro presentazione sono particolarmente rapidi a produrli in forma visibile per gli umani. Tuttavia hanno un paio di difetti tutt'altro che trascurabili.

Il primo difetto è che un'immagine di pochi pixel non può essere ingrandita su un'area che contenga più pixel, perché essa non contiene abbastanza informazione per colorare correttamente i pixel della riproduzione. Un'immagine quadrata di 100 pixel per 100 pixel, può essere rappresentata abbastanza bene sullo schermo di un PC ed apparirà un'immagine di circa 30 mm per 30 mm. Se si vuol vedere a pieno schermo la stessa immagine, cioè su un'area di 800 pixel

per 640 pixel, si capisce bene che il meglio che si possa fare è di rappresentare i pixel dell'immagine come quadrati di circa 8 pixel per 8 pixel, cioè sgranando l'immagine, cosicché appare come un mosaico di quadrati tutt'altro che piccoli, con la conseguenza che i contorni non sono più nitidi. La situazione peggiora notevolmente sulla carta dove il quadrato di 100 pixel per 100 pixel copre un quadratino di circa 8 mm per 8 mm stampando con una stampante con una densità di 300 puntini al pollice e di circa 4 mm per 4 mm stampando a 600 puntini al pollice; se si volesse ingrandire l'immagine come la si vede sullo schermo, l'effetto mosaico sarebbe talmente importante che difficilmente si riuscirebbe a riconoscere l'immagine.

La situazione è un poco migliore se si vuole ridurre l'immagine su un'area che contenga un po' meno pixel, ma il tutto è affidato alla qualità del tipo di rappresentazione o di stampa. Conclusione: si preferiscano sempre i formati vettoriali che possono essere ingranditi e rimpiccioliti a piacere. Tuttavia talvolta "bisogna fare di necessità virtù".

Se a questo inconveniente si aggiunge che alcuni formati sono compressi in una maniera che sfrutta la ridondanza delle immagini, ci si rende conto che un simile metodo perde parte dell'informazione e quando il file viene decompresso per la resa dell'immagine su schermo o su carta, appaiono degli artefatti che derivano dall'impossibilità di comprimere il file in modo decisivo senza perdere parte dell'informazione che l'immagine contiene. Se si tratta di una fotografia con colori sfumati e senza contorni netti, questo disturbo si nota poco, ma se si trattasse di un disegno al tratto la resa potrebbe essere inaccettabile.

Per le fotografie il formato `.jpg` va abbastanza bene ma può variare notevolmente da una fotografia all'altra. Per i disegni al tratto, se non fosse possibile avere dei disegni in formato vettoriale, allora il formato più adeguato sarebbe quello con estensione `.png`.

In ogni caso, fra i formati di questo paragrafo, quello di gran lunga preferibile sarebbe il formato `.pdf`, almeno nelle circostanze in cui il file PDF è stato composto senza perdere la natura vettoriale del disegno di partenza.

8.3 I formati accettabili a seconda del programma di composizione

A seconda che si usi LaTeX o pdfLaTeX alcuni formati sono accettabili e altri no. Peccato. Il solo formato accettabile da tutti sarebbe il formato di uscita di METAPOST. Ma anche per questo ci vuole un poco di maquillage. L'estensione di questi file è costituito da un numero progressivo, per esempio `.008` perché si trattava dell'ottavo disegno in uscita dallo stesso file sorgente. Questa estensione è accettabile da LaTeX ma non è accettabile da pdfLaTeX. Perciò è necessario cambiare l'estensione di tutti i file di uscita da METAPOST, aggiungendo l'estensione `.mps` o meglio cambiando il nome, per esempio da `mypicture.008` a `mypicture-008.mps`. Ciò fatto la cosa va bene sia per LaTeX sia per pdfLaTeX.

Una alternativa legata all'uso del pacchetto **graphics** potrebbe essere quella di definire una nuova regola di elaborazione dei file grafici; se si usa pdfLaTeX, si può usare il comando `\DeclareGraphicsRule` e specificare nel preambolo

```
\DeclareGraphicsRule{*}{mps}{}{}
```

Con questo si dichiara che qualunque altro formato (asterisco), diverso da quelli per i quali esiste una apposita regola, debba venire trattato come un file ottenuto da METAPOST. Questo è un escamotage che presenta qualche inconveniente, ma se si è sicuri di usare solo file ottenuti da METAPOST, non dovrebbe presentarsi nessun problema.

Tuttavia bisogna stare attenti a tutti gli altri formati secondo quanto detto qui di seguito.

8.3.1 I formati accettabili da LaTeX

LaTeX accetta solo i formati vettoriali; quindi oltre all'uscita di METAPOST, esso accetta solo i formati `.eps` e `.ps`; il formato Encapulated è preferibile rispetto al formato PostScript, perché sicuramente non contiene comandi PostScript che risultano incompatibili con l'inserimento delle figure all'interno di un'altra pagina PostScript.

8.3.2 I formati accettabili da pdfLaTeX

pdfLaTeX, oltre al formato di uscita di METAPOST e al formato `.pdf`, per ovvi motivi di compatibilità, accetta in entrata anche i formati '`.jpg` e `.png`.

I formati PostScript, spesso propagandati oltre misura, nonostante il loro indubbio vantaggio di essere vettoriali, *non sono accettabili da pdfLaTeX*.

8.4 Conversione dei formati

Ci troviamo forse in un vicolo cieco? Per fortuna no. Basta poter convertire da un formato all'altro. Esistono diversi programmi per farlo e qui se ne elencano alcuni; la maggior parte sono disponibili per tutte le piattaforme.

ghostscript e, meglio ancora, la sua interfaccia grafica ghostview (conosciuta anche col nome gview) permettono di trasformare una immagine `.ps` oppure `.eps` in immagine `.pdf` e viceversa. All'occorrenza il programma è capace di trasformare un file `.ps` in file `.eps` assicurando che il file finale non contenga istruzioni PostScript incompatibili con la possibilità di inclusione in un altro file.

epstopdf e ps2pdf, o altri simili applicativi da linea di comando, eseguono le analoghe trasformazioni senza interfaccia grafica; spesso hanno il vantaggio di corredare il file di uscita dell'informazione corretta in merito al 'bounding box', cioè alle dimensioni del rettangolo circoscritto all'immagine effettiva. Facendo uso del pacchetto **epstopdf** è possibile lanciare l'applicativo di conversione epstopdf direttamente durante l'esecuzione del programma di compilazione; affinché ciò sia possibile è necessario che il programma di compilazione sia abilitato con l'opzione della linea di comando `--shell-escape` che consente di eseguire programmi esterni. Come già detto altrove, questa è una pratica possibile, ma che apre una 'porta' a possibili attacchi 'virali', per cui se ne è sconsigliato l'uso; ovviamente in un contesto adeguatamente protetto questo pericolo non si pone o è estremamente improbabile.

eps2pdf è una interfaccia grafica disponibile solo per macchine Windows che esegue sostanzialmente lo stesso tipo di trasformazione di epstopdf, ma con il vantaggio dell'interfaccia grafica.

eps2png e eps2jpg sono programmi da linea di comando per macchine Linux/ UNIX e traformano le immagini vettoriali in formato `.eps` in immagini a matrici di punti diversamente compresse nei formati `.png` oppure `.jpg`. In generale sarebbe desiderabile evitare questi tipi di conversione, ma è opportuno limitarsi alle conversioni eseguibili con eps2pdf o epstopdf che conservano la qualità vettoriale dell'immagine.

pdftops esegue da linea di comando la conversione dal formato `.pdf` al formato `.ps`.

jpegtops trasforma un file `.jpg` in formato `.eps`. Si tratta di un applicativo da linea di comando. Assicura che il bounding box sia specificato correttamente. In realtà questo programma si limita ad avviluppare il codice compresso dell'immagine a matrice di pixel dentro un involucro PostScript, che contiene l'informazione corretta del 'bounding box', e a definire dove inizia e dove finisce il codice dell'immagine compressa.

Paint e i programmi simili aprono i file bitmapped e li trasformano in qualunque altro file a matrici di pixel, sia `.png` sia `.jpg`; quindi per gli altri formati bitmapped non dovrebbero esserci problemi per ottenere formati compatibili con i programmi di nostro interesse.

gimp si trova per tutte le piattaforme; è un formidabile programma di editing grafico che permette di eseguire tutte le trasformazioni dell'immagine e la loro conversione in un qualunque formato a matrici di pixel; riesce a farlo anche con input vettoriali, ma salva solo in formati a matrici di pixel. È un programma fortemente consigliabile per chiunque e su qualunque piattaforma. Sarebbe però consigliabile non usarlo per convertire formati vettoriali in formati a matrici di pixel, limitandosi alle altre trasformazioni e all'editing degli altri formati.

ImageMagik è un altro noto programma di editing grafico; esso mette a disposizione dell'utente il comando convert che permette di eseguire rapidamente la conversione da un formato grafico all'altro; viene consigliato (ed effettivamente viene installato automaticamente insieme alla distribuzione MacTEXdel sistema TEX sulle macchine Macintosh) e, per esempio, viene suggerito di definire la regola grafica

```
\DeclareGraphicsRule{.tif}{png}{.png}%
       {‘convert #1 ‘basename #1 .tif‘.png}
```

per convertire le immagini `.tif` in immagini `.png`. Questo metodo, in realtà, può venire usato su qualunque piattaforma e richiede l'abilitazione all'esecuzione di programmi esterni mediante l'opzione della linea di comando `--shell-escape`, della cui 'pericolosità' si è già parlato.

8.5 Scontornare le immagini

Un aspetto che si dimentica troppo spesso è quello di scontornare le immagini. LaTeX lascia da solo gli spazi bianchi necessari attorno alle immagini; se queste a loro volta contengono altri spazi bianchi al loro contorno, nel documento composto risulterà esserci alla fine troppo spazio bianco. Non è semplice poter scontornare le immagini, specialmente se ci si vuole attenere a programmi freeware.

gimp consente di scontornare e di correggere qualunque formato grafico ma permette di salvare solo nei formati a matrici di pixel; non è quindi il caso di servirsi di gimp per scontornare immagini vettoriali.

gview permette di correggere il bounding box, cosicché di fatto si scontorna l'immagine.

Adobe Acrobat non è freeware, ma permette di eseguire numerose azioni sui file e sulle immagini in formato PDF. Si possono per esempio estrarre delle pagine contenenti immagini da un file PDF e poi si possono estrarre, scontornando, solo le immagini che interessano andando a filo dell'immagine con una comoda interfaccia grafica. È possibile anche estrarre immagini in formato PostScript. L'unico difetto potrebbe essere il costo, visto che il programma è commerciale e il programma gratuito Adobe Reader permette di eseguire solamente la lettura ma non consente nessuna operazione di editing grafico.

Preview è un applicativo del sistema operativo Mac OS X; serve a molte funzioni, in particolare per visualizzare i file PDF; fra le operazioni che riesce a fare sui file PDF c'è anche l'operazione di *cropping*, cioè di scontornare. Ovviamente è usabile solo con quel sistema operativo.

Ma non tutto è perduto; se non si riesce a scontornare con programmi adeguati, si può sempre operare dall'interno di LaTeX o pdfLaTeX.

8.6 L'importazione delle immagini

Se il lettore non si è scoraggiato con queste (apparenti) difficoltà e ha già (all'occorrenza) convertito i formati delle immagini in uno dei formati consentiti, allora può procedere con l'effettiva importazione delle immagini.

Basta richiamare il pacchetto **graphicx** con il solito comando

```
\usepackage{graphicx}
```

e usarne i comandi e le opzioni in modo corretto.

Nel momento i cui si deve importare una figura esterna (della quale si assume di disporre del file nel formato corretto in relazione al programma usato) basta usare con la seguente sintassi il comando `\includegraphics` (generalmente ma non necessariamente all'interno dell'ambiente *figure*)

`\includegraphics[`⟨*lista delle chiavi*⟩`]{`⟨*file*⟩`}`

Il ⟨*file*⟩ è il nome del file contenente l'immagine *senza che ne venga specificata l'estensione*; se il formato giusto esiste, LaTeX o pdfLaTeX importa la figura; se il formato giusto non esiste emette un messaggio di errore e consente di proseguire, evidentemente senza importare nulla. Se si specifica l'estensione (magari errata) LaTeX o pdfLaTeX cerca solo quel file e non esplora la situazione possibile con altre estensioni e si lamenta con minacciosi messaggi di errore.

Se invece è stata definita una regola grafica, per esempio quella mostrata per convertire le immagini in formato `.tif` in formato `.png`, allora la prima volta che si esegue la compilazione viene ricercata anche l'estensione da convertire, viene eseguita la conversione e viene usato il file convertito; nelle eventuali compilazioni successive il programma trova direttamente il file nel formato giusto e non ha più bisogno di eseguire nessuna conversione.

Vale la pena di esporre un dettaglio organizzativo; per la gestione delle figure non ci sono problemi se i file che le contengono sono nella stessa cartella del file sorgente da comporre; tuttavia questo modo di procedere è disordinato e alla fine la cartella contiene tante cose disparate e non si riesce più a gestire. La soluzione è quella di dedicare/creare una sotto cartella dove custodire le immagini e solo loro. Supponendo di scrivere un libro su Paperino, Topolino e Pluto, probabilmente il file sorgente si troverà nella cartella `PTP/`; in questa cartella si trova la sotto cartella `foto/` (se non c'è la creiamo); in questa sotto cartella si trova la fotografia `Topolino.jpg`. Allora questa foto può essere inclusa per esempio specificando il percorso relativo per raggiungerla:

```
\includegraphics{foto/Topolino}
```

Più comodo e meno suscettibile di errori è invece il procedimento di specificare il percorso (*relativo alla cartella dove si trova il file sorgente*) dell'unica o delle varie cartelle dove si trovano le immagini mediante il comando `\graphicspath` con la seguente sintassi:

`\graphicspath{{`⟨$path_1$⟩`}{`⟨$path_2$⟩`}...{`⟨$path_n$⟩`}}`

dove è opportuno notare le parentesi graffe esterne che racchiudono le coppie di graffe interne, ognuna delle quali specifica un diverso percorso. Il comando

```
\graphicspath{{foto/}}
...
\includegraphics{Topolino}
```

Permette di includere la foto di Topolino senza bisogno di specificarne il percorso. Se le immagini relative a Paperino si trovassero in in un'altra sotto cartella, per esempio `fotoPaP/`, allora basterebbe specificare

```
\graphicspath{{foto/}{fotoPaP/}}
```

per avere accessibili entrambe le sotto cartelle senza bisogno di specificarle per ogni inclusione.

La ⟨*lista delle chiavi*⟩ è costituita da una serie di dichiarazioni separate da virgole e scritte nella forma 'chiave = valore'. Se la chiave rappresenta una affermazione booleana, non è necessario specificarne il valore 'true', ma, quando lo si vuole, è necessario specificarne il valore 'false'.

Qui non si esporranno tutte le opzioni disponibili; ci si limiterà ad descrivere quelle più comunemente usate, almeno secondo l'esperienza dello scrivente; si

rimanda il lettore alla documentazione del pacchetto per avere maggiori dettagli e per le altre opzioni.

`width` serve per specificare la larghezza dell'immagine nel documento finale; generalmente conviene parametrizzare questa dimensione ad un valore legato alla geometria della pagina, invece di specificare un numero di punti, o di millimetri, o di pollici, o di...[1]

Allora è meglio specificare la larghezza nella forma `width=0.5\linewidth` invece di `width=87mm`, perché cambiando layout della pagina l'informazione mantiene le proporzioni, mentre la dimensione assoluta potrebbe dare luogo a inconvenienti non lievi; si pensi per esempio di voler passare da una composizione a piena pagina ad una composizione a due colonne; 87 mm potrebbe andare bene a piena pagina, ma potrebbe essere più largo di una colonna.

Vale la pena di ricordare che `\linewidth` coincide con la giustezza corrente (piena pagina o colonna) salvo che all'interno di certi ambienti potrebbe essere un poco inferiore; è quindi conveniente fare sempre riferimento a `\linewidth` invece che a `\textwidth` o a `\columnwidth`.

Secondo una tradizione conservata dalle norme UNI, i fattori da preferire per moltiplicare la larghezza della linea sono 1, 0,5, 0,2 e le loro radici quadrate. Per le immagini i fattori più piccoli sono da evitare, ovviamente, ma la sequenza di ragione (approssimativa) $\sqrt{2}$ o $1/\sqrt{2}$ è quella più frequente.[2]

`height` serve per specificare l'altezza dell'immagine riprodotta nel documento; anche in questo caso è meglio fare riferimento all'altezza della pagina, parametrizzando rispetto a `\textheight`; per esempio, conservando uno dei fattori di scala della sequenza UNI, si potrebbe specificare: `height=0.35\textheight`.

`keepaspectratio` Se si sono specificate sia `width` sia `height` la figura potrebbe subire cambiamenti di scala diversi in altezza rispetto alla larghezza; per evitare questo fatto la variabile booleana `keepaspectratio` mantiene inalterato il rapporto di forma dell'immagine da riprodurre e sceglie sia per l'altezza sia per la larghezza il maggiore fra i due rapporti di scala che *non* fa eccedere nessuna delle due dimensioni specificate.

`viewport` serve per specificare la finestra attraverso cui guardare l'immagine; la finestra in effetti nasconde le parti dell'immagine che non interessano e può anche servire per scontornare una immagine alla quale non è stato possibile applicare il trattamento specificato nei paragrafi precedenti. Tuttavia può servire anche per mostrare solo un particolare dell'immagine. La sua sintassi è la seguente:

[1] Le unità di misura dimensionali che il sistema TeX capisce sono i punti (pt), i millimetri (mm), i centimetri (cm), i pollici (in) oltre alle unità legate alle dimensioni del font corrente; l'altezza della 'x' (ex), la larghezza di una 'M' (em). TeX capisce anche altre unità meno frequenti dall'uso un po' particolare, utili specialmente per chi scrive pacchetti di estensione o file di classe.

[2] Mescolando la sequenza: 0,5, 1, 2, con la sequenza: $\sqrt{0{,}5}$, $\sqrt{2}$ e arrotondando si ottiene la sequenza 0,35, 0,5, 0,7, 1, 1,4, 2, 2,8, 5.

`viewport=` ⟨*llx*⟩ ⟨*lly*⟩ ⟨*urx*⟩ ⟨*ury*⟩

dove ⟨*llx*⟩ ⟨*lly*⟩ sono le coordinate x e y dell'angolo in basso a sinistra della finestra e ⟨*urx*⟩ ⟨*ury*⟩ sono le coordinate x e y dell'angolo in alto a destra della finestra. Queste coordinate sono relative al rettangolo circoscritto all'immagine, e questo potrebbe non coincidere con il supporto virtuale sulla quale l'immagine digitale è riportata. Non è il caso di preoccuparsi di questo dettaglio perché il 99% delle volte si ha assoluta coincidenza.

`trim` serve ad una funzione simile a quella prodotta da `viewport`, solo che i quattro valori sono le ampiezze delle strisce di immagine da togliere successivamente da sinistra, dal basso, da destra e dall'alto dell'immagine; talvolta è più facile specificare questi valori, che non le coordinate della finestra attraverso cui vedere una parte dell'immagine. La sintassi è la seguente:

`trim=` ⟨*left*⟩ ⟨*bottom*⟩ ⟨*right*⟩ ⟨*top*⟩

Sia con la chiave `viewport` sia con `trim` le dimensioni sono espresse indicando le unità di misura che TEX gestisce; il programma provvede a trasformarle in *punti PostScript*, visto che l'operazione viene svolta in quel linguaggio vero o semplificato; se si desidera esprimere quelle lunghezze direttamente in punti PostScript si usa l'unità di misura `bp`, oppure non si specifica l'unità di misura che per default viene quindi assunta uguale a `bp`.

`clip` serve per dare l'ordine di tagliare effettivamente quanto sporge fuori del rettangolo specificato da `trim` o da `viewport`. Se non si esprimesse `clip`, il programma tratterebbe l'immagine come se le sue dimensioni effettive fossero quelle del rettangolo specificato con `trim` o `vewport` ma quanto sporge non verrebbe tagliato via e quindi l'immagine sarebbe ancora tutta integra ma più grande di quanto uno si aspetterebbe. Se invece di `\includegraphics` si usasse la versione asteriscata `\includegraphics*`, allora non sarebbe necessario specificare la parola chiave `clip`, perché il valore `true` con il comando asteriscato è quello di default.

`angle` serve per specificare l'angolo di rotazione (in gradi e in senso antiorario) della figura; quando si usa questa specifica (indicando generalmente $\pm 90°$ o $180°$) insieme a indicazioni di scala del tipo `width=...` bisogna stare attenti a quale operazione si indica per prima e perciò si esegue per prima; normalmente è conveniente specificare per prima la rotazione, perché è proprio quello che si vuole ottenere.

Bene inteso è possibile specificare anche altre chiavi e i loro valori, ma a quel punto è meglio andare a leggere direttamente la documentazione `grfguide.pdf` che accompagna sempre il sistema TEX e si trova nella cartella `.../doc/latex/graphics/`.

Nella figura 8.1 sono rappresentati alcuni effetti che si possono ottenere dalla stessa fotografia; sotto ogni foto è indicata la lista delle chiavi usata per ottenere il risultato. Nella prima foto in alto a sinistra si ha l'immagine a larghezza piena, fatta alla manifestazione Cheese, patrocinata da Slow Food, che si svolge

`width=\linewidth`

`height=.5\linewidth`

`width=\linewidth,`
`height=.5\linewidth`

`width=\linewidth,`
`height=0.5\linewidth,`
`keepaspectratio`

`height=.5\linewidth,`
`trim=100 50 150 120, clip`

`height=0.5\linewidth,`
`viewport=100 50 492 362, clip`

`width=\linewidth, angle=90`

`angle=90, width=\linewidth`

Figura 8.1: Due foto trattate con diverse chiavi

ogni due anni a Bra (CN); nelle immagini successive diverse chiavi sono state applicate e, in particolare, quando sono state specificate sia `width` sia `height` senza specificare `keepaspectratio`, si è ottenuta una deformazione dell'immagine in senso orizzontale che è decisamente visibile ad occhio nudo. Nelle due immagini dove si è specificato o `trim` o `viewport` insieme alla dichiarazione `clip` l'immagine risulta correttamente ritagliata e ingrandita fino a soddisfare il requisito dell'altezza. Nelle ultime immagini la foto del Minareto di Marrakesh "La Koutoubia" è stata ripresa con la fotocamera ruotata di 90° (be', ecco, ehm, circa...) per cui la corretta rappresentazione richiede una rotazione contraria; appare evidente che nella foto di sinistra la base è stata scalata fino ad essere uguale alla larghezza della riga, poi l'immagine è stata ruotata così che la base diventa l'altezza; invece nella foto di destra prima l'immagine è stata ruotata poi la base è stata scalata alla larghezza della riga e solo in questo modo si è ottenuto il risultato desiderato.

È bene notare che le operazioni eseguite con le chiavi `trim` e `viewport` non sembrano così semplici da fare perché non sono generalmente note le dimensioni naturali dell'immagine; tuttavia è bene ricordare che ogni volta che si lancia LaTeX o pdfLaTeX, viene prodotto un file con estensione `.log` e con il nome uguale a quello del file sorgente che si è appena compilato. In questo file, un normale file testuale contenente solo caratteri ASCII, quindi leggibile senza problemi con qualunque programma di elaborazione testi, quando viene importato per la prima volta un dato file grafico contenente un'immagine, ne viene scritto il tipo e ne vengono fornite le dimensioni 'base per altezza' in punti tipografici della 'bounding box'; con queste dimensioni è possibile fare le debite proporzioni di ciò che si desidera ritagliare via e quindi non è difficile trovare i valori numerici da usare; ricordiamoci infatti che le coordinate da specificare sono quelle del file di partenza, non del file riprodotto, quindi è necessario riferirsi alle coordinate del file di partenza.

Capitolo 9

LaTeX: la matematica semplice

9.1 Introduzione

La forza di LaTeX, che ne ha determinato la grandissima diffusione iniziale in ambito accademico e tecnico, prima di diffondersi in ambito letterario e umanistico, è proprio la professionalità con la quale compone la matematica; in questo primo capitolo ci soffermeremo sulle strutture matematiche più semplici e nel prossimo capitolo scenderemo nei dettagli per strutture matematiche più complesse, avvalendoci del pacchetto di estensione **amsmath**.

Non è sempre necessario ricorrere al pacchetto **amsmath**; dipende dal tipo di matematica che si vuole comporre. In questo capitolo, quindi, ci familiarizziamo solamente con la composizione della matematica, ma, anche se LaTeX, così com'è appena installato, è già in grado di comporre strutture matematiche piuttosto avanzate, rimandiamo al prossimo capitolo ciò che, pur essendo componibile con LaTeX senza estensioni, riesce molto più comodo da comporre con le estensioni fornite dal pacchetto **amsmath**.

9.2 I modi matematici

LaTeX compone la matematica sostanzialmente in quattro modi; se si andasse per il sottile si scoprirebbe che i modi sono otto, ma questo il lettore interessato lo può studiare nel TeXbook. I modi che interessano la quasi totalità degli utenti sono i seguenti:

1. Il modo testuale. LaTeX compone in questo modo quando scrive una espressione matematica in linea con il testo, come quando si specifica che la sezione aurea è $\varphi = (\sqrt{5}-1)/2$ e che questo numero gode della proprietà che $1/\varphi = 1+\varphi$. Se si vuole comporre in questo modo anche quando LaTeX di default comporrebbe in modo diverso bisogna dare l'istruzione `\textstyle`.

2. Il modo display. LaTeX compone la matematica in display quando compone le espressioni in linee a se stanti, staccate dal testo precedente e

seguente mediante spazi bianchi di ampiezza adeguata per 'mettere in mostra' l'espressione; per esempio

$$\varphi = \frac{\sqrt{5}-1}{2} \qquad \text{e} \qquad \frac{1}{\varphi} = 1+\varphi \qquad \text{(sezione aurea)}$$

Se si vuole comporre in questo modo anche quando LaTeX di default comporrebbe in modo diverso bisogna dare l'istruzione `\displaystyle`.

3. Il modo degli indici primi. LaTeX compone gli apici e i pedici (gli esponenti e i deponenti) di primo ordine in questo modo; usa un carattere di corpo più piccolo rispetto a quello delle variabili principali e li rialza per gli apici o li abbassa per i pedici di una quantità costante e prestabilita dalle caratteristiche dei font usati; per esempio in display si hanno le soluzioni di una equazione di secondo grado nella forma

$$x_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

Se si vuole comporre in questo modo anche quando LaTeX di default comporrebbe in modo diverso bisogna dare l'istruzione `\scriptstyle`.

4. Il modo degli indici secondi. LaTeX compone gli apici e i pedici (gli esponenti e i deponenti) di secondo ordine, cioè quelli degli apici e dei pedici di primo livello, con un carattere ancora più piccolo e adeguatamente rialzato o ribassato rispetto alle variabili a cui si riferiscono che sono già per loro conto composte con il font degli indici di primo livello; per esempio

$$V_{R_\mathrm{e}} = R_\mathrm{e} I_\mathrm{e}$$

Se si vuole comporre in questo modo anche quando LaTeX di default comporrebbe in modo diverso bisogna dare l'istruzione `\scriptscriptstyle`.

Di solito è piuttosto raro che si debba specificare lo stile, ovvero il modo di composizione matematica; è molto meglio lasciare fare a LaTeX che sa quasi sempre qual è il modo giusto da usare per la composizione.

Per formule così semplici come quelle esposte nell'enumerazione precedente basta che il compositore scriva il testo sorgente in lettere nello stesso modo come se lo dettasse al telefono (in inglese). Per esempio:

```
\begin{displaymath}
   x_{1,2}= \frac{-b\pm\sqrt{b^2-4ac}}{2a}
\end{displaymath}
```

L'ambiente *displaymath* dice "metti in display quanto racchiudo". Il segno `_` serve per dire "metti in posizione di deponente quanto contenuto nel prossimo gruppo"; il gruppo è delimitato dalle parentesi graffe, come si è già avuto modo di osservare; `\frac` sta per *fraction*; è una abbreviazione di una parola inglese, ma è una parola quasi completa; essa dà l'ordine di comporre una frazione; quindi bisogna specificare il numeratore e il denominatore mediante le rispettive espressioni racchiuse nei rispettivi gruppi; all'interno del numeratore compare il comando `\pm` che vuol dire "metti un segno 'più o meno'"; compare anche il comando `\sqrt`, sigla inglese che sta per *square root*, radice quadrata, ed esso

comanda di mettere sotto radice il contenuto del gruppo che lo segue; il segno ^ vuol dire "metti ad esponente il contenuto del prossimo gruppo", ma in questo caso il gruppo non è evidenziato con le graffe, perché è costituito da un solo token[1].

Come si vede la scrittura di una breve espressione è semplicissima e non richiede una memoria particolare per ricordare i comandi da usare. Quando ci sono da scrivere delle lettere greche esse si scrivono con i loro nomi per disteso (in inglese) e le maiuscole si scrivono con l'iniziale maiuscola; per non confondere le sequenze di lettere che formano i nomi delle lettere greche come semplici variabili matematiche scritte in lettere latine, i nomi delle lettere greche sono preceduti dal segno '\': `\alpha`, `\beta`, `\gamma`, ..., `\Gamma`, `\Delta`, eccetera.

Per gli operatori matematici più comuni bastano i tasti della tastiera di un qualunque PC o laptop. Se il PC o il laptop è dotato di una tastiera estesa o avanzata, è possibile comporre da tastiera anche altri segni che non sono serigrafati sui tasti; bisogna solamente premere contemporaneamente uno o più tasti di controllo (Alt, Shift, Ctrl, ecc.) e almeno un tasto 'ordinario'. In ogni caso nelle prossime pagine appaiono un certo numero di tabelle che contengono la lista completa dei comandi matematici di LaTeX che si possono introdurre senza disporre di nessuna tastiera speciale.

Gli spazi che in matematica compaiono nel file sorgente non hanno nessuna rilevanza; LaTeX mette gli spazi matematici dove sono necessari e, anche se offre dei comandi per inserire esplicitamente alcuni tipi di spazi matematici, il compositore è vivamente invitato a non farne uso! LaTeX compone benissimo da solo, senza che il compositore debba perdere tempo per rendere il 99% delle volte il frutto della composizione meno bello di quello che LaTeX avrebbe fatto da solo.

Esistono alcuni pochi casi ben noti in cui il compositore può inserire degli spazi matematici; essi riguardano tutti gli operatori obliqui quando gli oggetti che li precedono o li seguono hanno forme particolari, oppure quando il compositore deve lavorare con integrali multipli e non vuole usare il pacchetto di estensione **amsmath**, di cui si parlerà nel prossimo capitolo. Quindi il compositore sarà autorizzato a correggere gli spazi usati da LaTeX solo dopo aver corretto le bozze e se vuol aggiungere quei pochi e rari tocchi di qualità che LaTeX non è riuscito a mettere da solo.

Per aiutare il compositore ad avere sottomano tutti i simboli che LaTeX mette a disposizione (e sono moltissimi) sono state preparate alcune tabelle di segni accostati ai comandi che li producono; si vedano allora le tabelle 9.1–9.8

Va detto subito che nelle tabelle 9.2, 9.3 e 9.8 i segni con uno sfondo grigio non sono usabili se non si usa il pacchetto di estensione **amsmath** oppure il pacchetto di estensione **latexsym**; in realtà questo secondo pacchetto rende accessibili alcuni segni che con il vecchio LaTeX 209 erano direttamente accessibili senza estensioni; perciò questo pacchetto è da considerare come una maniera per compilare vecchi file sorgente predisposti per il vecchio LaTeX. Con il nuovo LaTeX 2_ε questi segni sono tutti accessibili tramite il pacchetto **amsmath**. Siccome il pacchetto **amsmath** è utilissimo per la composizione di testi tecnici che fanno uso di molta matematica, questo pacchetto viene solitamente caricato comunque, e allora tanto vale caricare solo quello. Tuttavia bisogna ancora

[1] I comandi `\[` e `\]` sostituiscono efficacemente i comandi di apertura e di chiusura dell'ambiente *displaymath*.

Minuscole					
\alpha	α	\iota	ι	\rho	ρ
\beta	β	\kappa	κ	\sigma	σ
\gamma	γ	\lambda	λ	\tau	τ
\delta	δ	\mu	μ	\upsilon	υ
\epsilon	ϵ	\nu	ν	\phi	ϕ
\zeta	ζ	\xi	ξ	\chi	χ
\eta	η	o	o	\psi	ψ
\theta	θ	\pi	π	\omega	ω
Varianti delle minuscole					
\varepsilon	ε	\varpi	ϖ	\varsigma	ς
\vartheta	ϑ	\varrho	ϱ	\varphi	φ
Maiuscole					
\Gamma	Γ	\Xi	Ξ	\Phi	Φ
\Delta	Δ	\Pi	Π	\Psi	Ψ
\Theta	Θ	\Sigma	Σ	\Omega	Ω
\Lambda	Λ	\Upsilon	Υ		
Maiuscole corsive					
\mathit{\Gamma}	$\mathit{\Gamma}$	\mathit{\Xi}	$\mathit{\Xi}$	\mathit{\Phi}	$\mathit{\Phi}$
\mathit{\Delta}	$\mathit{\Delta}$	\mathit{\Pi}	$\mathit{\Pi}$	\mathit{\Psi}	$\mathit{\Psi}$
\mathit{\Theta}	$\mathit{\Theta}$	\mathit{\Sigma}	$\mathit{\Sigma}$	\mathit{\Omega}	$\mathit{\Omega}$
\mathit{\Lambda}	$\mathit{\Lambda}$	\mathit{\Upsilon}	$\mathit{\Upsilon}$		

Tabella 9.1: Le lettere greche

segnalare che alcuni di questi simboli, come per esempio υ, non dovrebbero mai venire usati perché contrari alle norme ISO, quindi non è una gran perdita se non se ne fa uso.

9.3 Alcune annotazioni sulle lettere greche

Per quanto riguarda le lettere greche, si può notare che alcune di esse hanno delle forme varianti i cui nomi cominciano con l'abbreviazione `var`. Esse sono delle varianti per quel che riguarda gli Stati Uniti, ma, tolto ϖ che non si usa mai, forse nemmeno negli Stati Uniti, gli altri segni sono quelli usati normalmente in Europa e in Italia; in alcuni file di classe che lo scrivente ha creato, egli ha scambiato i nomi varianti con quelli ordinari per rendere normali quelli che vengono usati di solito in Europa.

È inoltre consuetudine in tutto il mondo di scrivere le lettere greche maiuscole con caratteri diritti e non inclinati, nonostante le norme ISO prescrivano l'uso delle lettere inclinate per qualsiasi variabile matematica o fisica.

Se si desidera scrivere le lettere greche maiuscole con caratteri inclinati bisogna fare esattamente come è suggerito nella tabella 9.1. Oppure, se si sta facendo uso del pacchetto **amsmath**, si ottengono le lettere inclinate usando i comandi varianti (non visibili in nessuna delle tabelle citate) dove il nome delle lettere greche con iniziale maiuscola è preceduto da `var`. Scrivendo quindi `\varGamma`, `\varDelta`, `\varTheta`, eccetera, si ottengono i simboli $\varGamma$, $\varDelta$, $\varTheta$, ...

\approx	$\approx$	\asymp	$\asymp$	\bowtie	$\bowtie$
\cong	$\cong$	\dashv	$\dashv$	\doteq	$\doteq$
\downarrow	$\downarrow$	\Downarrow	$\Downarrow$	\equiv	$\equiv$
\frown	$\frown$	\ge	$\geq$	\geq	$\geq$
\gets	$\gets$	\gg	$\gg$	\hookleftarrow	$\hookleftarrow$
\hookrightarrow	$\hookrightarrow$	\iff	$\iff$	\in	$\in$
\Join	$\bowtie$	\le	$\leq$	\leadsto	$\leadsto$
\leftarrow	$\leftarrow$	\Leftarrow	$\Leftarrow$	\leftharpoondown	$\leftharpoondown$
\leftharpoonup	$\leftharpoonup$	\leftrightarrow	$\leftrightarrow$	\Leftrightarrow	$\Leftrightarrow$
\leq	$\leq$	\ll	$\ll$	\longleftarrow	$\longleftarrow$
\Longleftarrow	$\Longleftarrow$	\longleftrightarrow	$\longleftrightarrow$	\Longleftrightarrow	$\Longleftrightarrow$
\longmapsto	$\longmapsto$	\longrightarrow	$\longrightarrow$	\Longrightarrow	$\Longrightarrow$
\mapsto	$\mapsto$	\mid	$\mid$	\models	$\models$
\ne	$\neq$	\nearrow	$\nearrow$	\neq	$\neq$
\ni	$\ni$	\not=	$\neq$	\nwarrow	$\nwarrow$
\parallel	$\parallel$	\perp	$\perp$	\prec	$\prec$
\preceq	$\preceq$	\propto	$\propto$	\rightarrow	$\rightarrow$
\Rightarrow	$\Rightarrow$	\rightharpoondown	$\rightharpoondown$	\rightharpoonup	$\rightharpoonup$
\rightleftharpoons	$\rightleftharpoons$	\searrow	$\searrow$	\sim	$\sim$
\simeq	$\simeq$	\smile	$\smile$	\sqsubset	$\sqsubset$
\sqsubseteq	$\sqsubseteq$	\sqsupset	$\sqsupset$	\sqsupseteq	$\sqsupseteq$
\subset	$\subset$	\subseteq	$\subseteq$	\succ	$\succ$
\succeq	$\succeq$	\supset	$\supset$	\supseteq	$\supseteq$
\swarrow	$\swarrow$	\to	$\to$	\uparrow	$\uparrow$
\Uparrow	$\Uparrow$	\vdash	$\vdash$		

Tabella 9.2: Gli operatori di relazione

\amalg	$\amalg$	\ast	$\ast$	\bigcirc	$\bigcirc$
\bigtriangledown	$\bigtriangledown$	\bigtriangleup	$\bigtriangleup$	\bullet	$\bullet$
\cap	$\cap$	\cdot	$\cdot$	\circ	$\circ$
\cup	$\cup$	\dagger	$\dagger$	\ddagger	$\ddagger$
\diamond	$\diamond$	\div	$\div$	\lhd	$\lhd$
\mp	$\mp$	\odot	$\odot$	\ominus	$\ominus$
\oplus	$\oplus$	\oslash	$\oslash$	\otimes	$\otimes$
\pm	$\pm$	\rhd	$\rhd$	\setminus	$\setminus$
\sqcap	$\sqcap$	\sqcup	$\sqcup$	\star	$\star$
\times	$\times$	\unlhd	$\unlhd$	\unrhd	$\unrhd$
\uplus	$\uplus$	\vee	$\vee$	\wedge	$\wedge$
\wr	$\wr$				

Tabella 9.3: Gli operatori binari

Operatori senza limiti				
\arccos	\arcsin	\arctan	\arg	\cos
\cosh	\cot	\coth	\csc	\deg
\dim	\exp	\hom	\ker	\lg
\ln	\log	\sec	\sin	\sinh
\tan	\tanh			
Operatori con limiti				
\det	\gcd	\inf	\lim	\liminf
\limsup	\max	\min	\Pr	\sup

Tabella 9.4: Gli operatori funzionali

\bigcap	$\bigcap$	$\cap$	\bigvee	$\bigvee$	$\vee$
\bigcup	$\bigcup$	$\cup$	\bigwedge	$\bigwedge$	$\wedge$
\bigodot	$\bigodot$	$\odot$	\coprod	$\coprod$	$\amalg$
\bigoplus	$\bigoplus$	$\oplus$	\int	$\int$	$\int$
\bigotimes	$\bigotimes$	$\otimes$	\oint	$\oint$	$\oint$
\bigsqcup	$\bigsqcup$	$\sqcup$	\prod	$\prod$	$\prod$
\biguplus	$\biguplus$	$\uplus$	\sum	$\sum$	$\sum$

\sqrt	$\sqrt{}$	$\sqrt{}$	$\sqrt{}$	$\sqrt{}$	$\sqrt{}$

Tabella 9.5: I grandi operatori

(	/ 	\langle 	\rangle
) 	\lfloor	\rfloor	\lgroup
[	\lceil 	\rceil 	\rgroup 
] 	\backslash 	\vert 	\bracevert 
\\| 	\Vert 	\updownarrow 	\Updownarrow 
| 	\uparrow 	\Uparrow 	\downarrow 
} 	\arrowvert 	\Arrowvert 	\Downarrow 
{ 	\lmoustache 	\rmoustache 	

Tabella 9.6: I grandi delimitatori

\hat{a}	$\hat{a}$	\acute{a}	$\acute{a}$	\bar{a}	$\bar{a}$	\dot{a}	$\dot{a}$
\check{a}	$\check{a}$	\grave{a}	$\grave{a}$	\vec{a}	$\vec{a}$	\ddot{a}	$\ddot{a}$
\breve{a}	$\breve{a}$	\tilde{a}	$\tilde{a}$	\widehat{AB}	$\widehat{AB}$	\widehat{ABC}	$\widehat{ABC}$
\mathring{a}	$\mathring{a}$			\widetilde{AB}	$\widetilde{AB}$	\widetilde{ABC}	$\widetilde{ABC}$

Tabella 9.7: Gli accenti matematici

\aleph	$\aleph$	\angle	$\angle$	\backslash	$\backslash$
\bot	$\bot$	\Box	$\Box$	\clubsuit	$\clubsuit$
\Diamond	$\Diamond$	\diamondsuit	$\diamondsuit$	\ell	ℓ
\emptyset	$\emptyset$	\exists	$\exists$	\flat	$\flat$
\forall	$\forall$	\hbar	$\hbar$	\heartsuit	$\heartsuit$
\Im	$\Im$	\imath	$\imath$	\infty	∞
\jmath	$\jmath$	\mho	$\mho$	\nabla	∇
\natural	$\natural$	\neg	$\neg$	\partial	∂
\prime	$\prime$	\Re	$\Re$	\sharp	$\sharp$
\spadesuit	$\spadesuit$	\surd	$\surd$	\top	$\top$
\triangle	$\triangle$	\wp	$\wp$	\\|	$\|$

Tabella 9.8: Altri simboli

9.4 Alcune osservazioni sugli operatori funzionali

Gli operatori funzionali come sin, arctan, eccetera sono presentati nella tabella 9.4; essi *devono* essere usati esattamente con quei comandi per due importanti motivi.

1. I nomi che vengono scritti nel documento durante l'esecuzione di quei comandi sono quelli prescritti dalle norme internazionali ISO e, siccome essi sono ripresi dalle norme nazionali UNI e queste sono legge dello Stato, non è consentito usare nomi diversi, anche se si è abituati a scrivere 'tg' al posto di 'tan' oppure 'sen' al posto di 'sin'.

2. Gli operatori, in quanto tali, richiedono certi spazi alla loro destra e alla loro sinistra; questi spazi dipendono dalla natura degli oggetti matematici che precedono o seguono gli operatori; solo usando quei comandi LaTeX sa che sta usando degli operatori e sa quali spazi usare.

La tabella 9.4 suddivide gli operatori in due categorie: (*a*) quelli chiamati 'operatori senza limiti' nei quali le indicazioni di esponente o di deponente vengono usate normalmente come tali, e (*b*) quelli chiamati 'operatori con limiti' per i quali le indicazioni di esponente e di deponente vengono usati come nei simboli di sommatoria; si noti la differenza nell'equazione 9.1

$$\lim_{x\to 0}\sum_{k=0}^{\infty}\sin^2 2kx = 0 \tag{9.1}$$

scritta facendo uso dei segni `_` e `^` che normalmente indicano rispettivamente l'inserzione di un deponente o di un esponente:

```
\lim_{x\to 0}\sum_{k=0}^\infty \sin^2 2kx = 0
```

Si noti che in modo testo LaTeX compone i 'limiti' degli operatori con limiti, delle sommatorie, degli integrali, e simili, come se fossero dei deponenti o degli esponenti; esso si comporta in questo modo per evitare di far diventare troppo grandi (alte e profonde) le righe con gli operatori con limiti che obbligherebbero

all'inserimento di una generosa interlinea; in modo display, invece tutto procede come esemplificato nell'equazione 9.1.

Se si desidera definire nuovi operatori funzionali oltre a quelli che compaiono nella tabella 9.4, il pacchetto **amsmath** mette a disposizione due comandi con la seguente sintassi:

```
\DeclareMathOperator{⟨operatore⟩}{⟨definizione⟩}
\DeclareMathOperator*{⟨operatore⟩}{⟨definizione⟩}
```

dove ⟨*operatore*⟩ è il comando con il quale si vuole invocare l'operatore mentre la ⟨*definizione*⟩ contiene il modo di scrivere il nome dell'operatore. La dichiarazione asteriscata definisce il comando per un operatore con limiti, mentre quella non asteriscata lo definisce per un operatore senza limiti.

Per esempio, l'operatore per separare la parte reale di un numero complesso è `\Re` e produce il simbolo $\Re$; volendolo ridefinire in modo che esso sia scritto in caratteri non gotici, ma sia in neretto, si potrebbe usare la dichiarazione[2]

```
\DeclareMathOperator{\Re}{\mathbf{Re}}
```

In questo modo lo stesso comando `\Re` non produce più $\Re z = \Re(x + \mathrm{i}\, y) = x$ ma $\mathbf{Re}\, z = \mathbf{Re}(x + \mathrm{i}\, y) = x$ e tratta il nuovo simbolo come un operatore senza limiti lasciandovi attorno gli stessi spazi che lascia normalmente attorno agli operatori. Si noti che anche l'unità immaginaria 'i' è un operatore (vettoriale) ed è stato stampato facendo uso del comando `\uimm` la cui definizione un po' particolare verrà discussa nella pagina 110.

9.5 Alcune osservazioni sui grandi operatori

I grandi operatori hanno almeno due dimensioni, una 'normale' adatta al modo testo e una più grande adatta al modo display; la radice quadrata, invece ha numerose varianti adatte alla grandezza del radicando.

Così in modo testo si potrà scrivere con l'operatore piccolo (scelto automaticamente da LaTeX senza che il compositore se ne debba preoccupare) $\sum_{k=0}^{\infty} x^k = 1/(1-x)$ mentre in display sarà

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \tag{9.2}$$

Bisogna fare attenzione ad un dettaglio non trascurabile. Mentre la maggior parte delle volte LaTeX usa correttamente (almeno nel senso delle nostre aspettative) i grandi operatori scegliendo l'operatore grande oppure quello piccolo, talvolta ci sembra che si sbagli; si osservi l'equazione 9.3:

$$\frac{1}{\sum_{k=0}^{\infty} x^k} = 1 - x \tag{9.3}$$

Si nota che, nonostante l'equazione sia in display, LaTeX ha usato l'operatore del modo testo. A noi sembra scorretto, ma LaTeX ha agito correttamente, perché,

[2] Il comando `\Re` è già definito e i vari comandi `\Declare...` non producono messaggi d'errore se si ridefiniscono comandi esistenti; controllano però l'esistenza di un comando preesistente e, nel caso, scrivono un avvertimento nel file `.log`.

nonostante si trovi a comporre una formula in display, se usasse la forma grande dell'operatore, l'intera formula risulterebbe squilibrata

$$\frac{1}{\sum_{k=0}^{\infty} x^k} = 1 - x \tag{9.4}$$

come si vede bene nell'equazione 9.4.

LaTeX si comporta in questo modo tutte le volte che cerca di mantenere più o meno costante l'altezza delle righe nelle espressioni matematiche; esso infatti passa al modo testo anche per scrivere gli elementi delle matrici; bisogna ricordarsene e, se non ci piacesse, sarebbe nostra cura modificare il modo di scrivere le formule in modo da ottenere una forma grafica gradevole e nello stesso tempo corretta; si veda a questo proposito l'equazione 9.6.

9.6 I grandi delimitatori

Nella tabella 9.6 sono rappresentati i grandi delimitatori, o meglio i delimitatori estensibili; questi si possono estendere in altezza in modo virtualmente illimitato e possono racchiudere espressioni di qualunque grandezza; si pensi anche solo ai delimitatori che differenziano una matrice da un determinante. Ma i delimitatori possono essere resi estensibili solo premettendo al delimitatore di sinistra il comando `\left` e al delimitatore di destra il comando `\right`; questi due comandi devono essere sempre appaiati nella stessa espressione matematica, quindi se si deve mettere un solo delimitatore, l'altro lo si rende invisibile scrivendovi un 'punto' invece del segno di una parentesi; si veda l'equazione 9.5

$$x_{1,2} = \begin{cases} \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a} & \text{se} \quad b^2 - 4ac > 0 \\ -\dfrac{b}{2a} & \text{se} \quad b^2 - 4ac = 0 \\ \dfrac{-b \pm \mathrm{i}\sqrt{4ac - b^2}}{2a} & \text{se} \quad b^2 - 4ac < 0 \end{cases} \tag{9.5}$$

composta scrivendo

```
x_{1,2} = \left\{ ... \right.
```

Per comporre la struttura dell'equazione 9.5, già più complessa di quelle usate come esempio fino ad ora, si può usare l'ambiente *cases* fornito dal pacchetto **amsmath**; con questo ambiente non è il caso di preoccuparsi di usare delimitatori invisibili perché ci pensa lui; anzi, questa osservazione ci fa capire meglio perché è meglio servirsi di ambienti di composizione appositamente predisposti dal file di classe o dai pacchetti di estensione o da definizioni create dal compositore, piuttosto che ripetere ogni volta l'intero assemblaggio di comandi e di istruzioni, col rischio di dimenticarne qualcuno o di non essere coerenti ogni volta che si dovrebbe comporre lo stesso genere di struttura nella stessa maniera.

Ma `\left` e `\right` sono comodi anche per scrivere correttamente in modo display una formula come l'equazione 9.4 che diventa:

$$\left(\sum_{k=0}^{\infty} x^k \right)^{-1} = 1 - x \tag{9.6}$$

Bisogna prestare una particolare attenzione quando si deve spezzare una equazione fra più righe; quando si è in modo testo si suppone che le espressioni stampate siano molto semplici e non troppo lunghe; se l'espressione cade in fin di riga LaTeX (con molta riluttanza) provvede a dividere l'espressione dopo un operatore di relazione o dopo un operatore binario. Tuttavia sta al compositore evitare che si debba andare a capo mentre si è all'interno di una parentesi. Quando invece si è in display le espressioni possono essere anche molto lunghe e potrebbe essere necessario andare a capo all'interno di una parentesi. Solo chi ha scritto l'equazione ne conosce il ritmo di lettura e solo lui sa dove è meno grave andare a capo. Tuttavia resta la regola principale che si può andare a capo dopo gli operatori di relazione e, subordinatamente, dopo gli operatori binari, mai dopo gli operatori funzionali o i grandi operatori o immediatamente dopo i delimitatori di apertura. Ecco perché sono utili le tabelle 9.3–9.6, così anche chi non è matematico sa di che cosa si sta parlando.

Nella disgraziata ipotesi che fosse necessario andare a capo all'interno di una lunga espressione fra parentesi, allora potrebbero risultare utili i seguenti comandi, che, come `\left` e `\right` ingrandiscono i delimitatori, ma di una quantità fissa, indipendentemente da che cosa essi delimitino; non è difficile ottenere

$$\Biggl(\biggl(\Bigl(\bigl(()\bigr)\Bigr)\biggr)\Biggr)$$

con i comandi

```
\Biggl(\biggl(\Bigl(\bigl(()\bigr)\Bigr)\biggr)\Biggr)
```

La lettera `l` o `r` alla fine di ogni comando specifica se il segno che segue è un delimitatore di sinistra (*left*) o di destra (*right*); esisterebbe anche un suffisso `m` per i separatori che talvolta si trovano a metà di certe espressioni matematiche.

9.7 Gli accenti matematici

Per segnare con 'diacritici' matematici i vari simboli che si usano esistono i comandi riportati nella tabella 9.7; sarebbe un grave errore usare quegli accenti per inserire diacritici in modo testo. Per questi si veda in appendice la tabella E.1. Per mettere accenti sulle lettere 'i' e 'j' in matematica bisogna ricorrere a due particolari versioni di questi segni privi del puntino: `\imath` e `\jmath`. Usando questi segni si può scrivere per esempio $\vec{a} = a_x\vec{\imath} + a_y\vec{\jmath}$.

9.8 Gli ambienti matematici

Per inserire una espressione matematica in linea con il testo bisogna entrare in modo matematico; la maniera più semplice, ma da evitare, è quella di racchiudere i comandi matematici all'interno di un gruppo costituito da due segni di dollaro `$`. Così per scrivere che $\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$ basta inserire nel testo sorgente

```
$\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta $
```

Ci si rende conto, però, che il segno di dollaro $ funziona come un interruttore che cambia modo da testo a matematico o viceversa, senza preoccuparsi di controllare se l'operazione sia corretta; LaTeX offre il modo di eseguire questo controllo se si racchiude l'espressione matematica fra i comandi \(e \), così:

```
\(
   \sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta
\)
```

Non è necessario andare a capo dopo \(e dopo la fine dell'espressione matematica, ma il testo sorgente è più facilmente leggibile.

Il comando \(può essere dato solo mentre si è in modo testo e \) solo quando si è in modo matematico; se ciò non succedesse, LaTeX provvederebbe ad emettere gli opportuni messaggi d'errore che consentirebbero di eseguire le necessarie correzioni con facilità.

Analogamente in modo display sarebbe possibile usare i comandi di basso livello costituiti da due coppie di segni di dollaro; per scrivere

$$\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$$

basterebbe scrivere nel file sorgente

```
$$
   \sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta
$$
```

ma di nuovo mancherebbe la diagnostica di LaTeX. Questo invece offre due ambienti per scrivere espressioni senza la numerazione e un ambiente per scrivere espressioni numerate:

```
\[
   \sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta
\]
```

con la sintassi

\[⟨*espressione matematica*⟩\]

oppure con l'ambiente *displaymath*

```
\begin{displaynmath}
   \sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta
\end{displaymath}
```

che segue la sintassi:

\begin{displaymath}
⟨*espressione matematica*⟩
\end{displaymath}

Mentre per le espressioni numerate si usa l'ambiente *equation* con la sintassi:

\begin{equation}
⟨*espressione matematica*⟩
\label{⟨*etichetta*⟩}
\end{equation}

Usando il pacchetto **amsmath** diventa disponibile l'ambiente *equation**

```
\begin{equation*}
⟨espressione matematica⟩
\end{equation*}
```

del tutto equivalente all'ambiente *displaymath* solo che accetta anche espressioni matematiche più complesse disponibili con quel pacchetto.

La sintassi è auto esplicativa, ma si noti che l'⟨*etichetta*⟩ può contenere un nome simbolico con il quale richiamare l'equazione per nome, invece che per numero, rendendo così automatico il processo dei riferimenti incrociati mediante i comandi `\label`, `\ref` e `\pageref`. Il pacchetto **amsmath** mette a disposizione anche il comando per citare le equazioni `\eqref` che provvede da solo a racchiuderne il riferimento vero fra parentesi tonde come si fa abitualmente (e come *non* si è fatto in questo testo). Quel pacchetto mette a disposizione anche un comando `\tag` per identificare le equazioni; esso consente di assegnare all'equazione un 'nome' letterale, invece che un identificativo numerico, ed è usabile anche all'interno dell'ambiente *equation**, benché questo si riferisca ad un ambiente non numerato; è proprio quello che si è fatto per l'equazione che definisce la sezione aurea nella pagina 86.

9.9 Le unità di misura

Quanto esposto nei paragrafi precedenti dovrebbe consentire di scrivere equazioni di una riga, numerate o non numerate e di poterle citare in modo corretto senza scambi di identità.

Prima di passare al capitolo successivo vale la pena di soffermarci su una questione molto importante, visto che la matematica non serve solo per i matematici puri, ma anche ai tecnologi, ai fisici, ai chimici, e a tanti altri tecnici e scienziati che scrivono equazioni di fisica–matematica.

La fisica–matematica usa espressioni che coinvolgono grandezze fisiche e con queste la composizione della matematica deve soddisfare a ulteriori regole.

1. Le grandezze fisiche vanno scritte in corsivo matematico; anche le costanti fisiche vanno scritte in corsivo matematico, a differenza delle costanti matematiche che vanno scritte in tondo;

2. quando si esplicita il valore numerico delle costanti fisiche o di qualunque grandezza fisica, bisogna inserire anche le unità di misura.

3. Le unità di misura devono seguire scrupolosamente le norme ISO, sia per i loro nomi, sia per i loro simboli e i loro prefissi decimali; inoltre

   (a) devono essere scritte in tondo anche nelle espressioni matematiche;

   (b) devono essere separate da uno spazio fine dal valore numerico;

   (c) questo spazio deve essere 'non separabile' nel senso che non si può avere un 'a capo' fra la misura e l'unità di misura;

   Tutte queste azioni si ottengono in un colpo solo se si usa il comando `\unit` disponibile con il pacchetto **babel** quando si scrive in italiano. Bisogna solo

Figura 9.1: Due esempi di strafalcioni giornalistici

scrivere il comando `\unit` con il suo argomento 'attaccato' alla misura, per esempio `3,9\unit{k\ohm}` che produce 3,9 kΩ.[3]

4. I pedici e gli apici numerici vanno scritti con carattere diritto, ma quelli letterali vanno scritti in corsivo matematico se rappresentano quantità variabili, ma vanno scritti in tondo se rappresentano delle apposizioni della grandezza fisica; quindi $V_{\max}$ e non V_{max}; invece bisogna scrivere V_T se T è, per esempio, una temperatura; V_i se i rappresenta l'i-esimo elemento di un insieme numerabile, ma V_{i} se 'i' sta per 'iniziale'. Per questo scopo **babel** in italiano offre il comando `\ped` per scrivere pedici in tondo; e `\ap` per scrivere apici in tondo; entrambi i comandi funzionano anche in modo testo per inserire pedici o apici, ma in modo testo mantengono il font in uso, senza preoccuparsi se sia diritto o inclinato.

Una tipica equazione fisica potrebbe essere

$$\tilde{V}(t) = \sqrt{2} V_{\mathrm{eff}}\, \mathrm{e}^{\mathrm{i}\,\omega t}$$

dove 'eff' sta per 'efficace', 'e' è la costante matematica base dei logaritmi naturali da distinguere dalla 'carica elettrica elementare e', e 'i' è l'unità immaginaria da distinguere dalla grandezza fisica 'corrente elettrica i'.

La figura 9.1 riprende due strafalcioni giornalistici pubblicati su un quotidiano nazionale; nella figura di sinistra c'è l'errore frequentissimo di scrivere 'K', invece di 'k' per il prefisso 'kilo'; in effetti in quella figura si parla di kelvin metri all'ora... Nella figura a destra 'M/bit al secondo' doveva essere[4] Mb/s o forse Mib/s. Ammesso che si tratti solo di un errore di stampa, con una velocità di trasmissione di 75 Mib/s per trasmettere il contenuto di un DVD completo (che generalmente contiene più di 4,5 GiB, cioè più di 36 Gib) occorrono almeno 480 s cioè 8 min, tutt'altro che pochi secondi...

Chi compone i suoi testi con LaTeX è tenuto ad astenersi da una simile ignoranza e sciatteria!

[3] In modo matematico bisognerebbe scrivere `\Omega`, mentre in modo testo bisognerebbe scrivere `\textohm`. Non è difficile definire un comando `\ohm` che distingua che cosa si deve usare a seconda se si è in modo matematico o in modo testo, come è stato fatto in questo documento; vedi nella pagina 185.

[4] I prefissi stabiliti dalle norme ISO dal kilo in poi rappresentano potenze di 1000. In informatica è più conveniente fare riferimento alle potenze di 2; per indicare i multipli normalmente si usano gli stessi prefissi ISO (con k mutato in K) seguiti da una 'i'; essi rappresentano le potenze di $2^{10} = 1024$. Il prefisso 'Mi' indica quindi $(2^{10})^2$ e il prefisso 'Gi' indica il valore $(2^{10})^3$. Inoltre il simbolo del bit è 'b', mentre quello del byte è 'B'.

Egli, però, può trarre giovamento dall'uso dei pacchetti **units** e **SIunits**. Il primo pacchetto definisce una versione del comando `\unit` (diversa da quella che è definita con l'opzione *italian* del pacchetto **babel**) che consente di specificare alcune opzioni di spaziatura fra la misura e l'unità di misura e consente di scrivere le frazioni mediante numeratore rialzato e denominatore ribassato, utili sia in linea con il testo, sia in matematica, affinché risulti ben chiaro chi sia il numeratore e chi il denominatore; si può quindi specificare una velocità nella forma $70\,{}^{\mathrm{km}}/_{\mathrm{h}}$; questo modo di scrivere è una pratica diffusa nei paesi dell'Europa del Nord, ma non è previsto dalle norme ISO.

Il secondo pacchetto contiene una bella e aggiornata descrizione della situazione relativa al Sistema Internazionale (SI) di unità di misura e richiama anche le norme tipografiche specificate dalla Conferenza Generale dei Pesi e delle Misure e dalle sue commissioni, nonché dalle norme ISO che regolano questa questione. Il pacchetto fornisce anche una diversa definizione del comando `\unit` che richiede però due argomenti e quindi è incompatibile con la definizione dell'opzione *italian* di **babel**; all'interno del secondo argomento è possibile specificare le unità di misura a parole (in inglese britannico e al singolare) per cui è possibile indicare la velocità di 70 km/h con la scrittura

```
\unit{70}{\kilo\metre\per\hour}
```

Invece con il comando definito con l'opzione *italian* di **babel** basta scrivere `70\unit{km/h}`.

Usando il pacchetto **SIunits** è possibile specificargli l'opzione *italian* cosicché la definizione di **babel** `\unit` diventa `\unita` (in italiano e senza accento) e si elimina ogni interferenza. Viceversa, se questa soluzione non interessa, basta caricare il pacchetto **babel** *dopo* il pacchetto **SIunits** per evitare che **babel** ridefinisca il comando `\unit` di **SIunits**.

Il compositore stia bene attento alla preposizione 'per', che in italiano significa 'moltiplicato' e in inglese significa 'diviso'; una resistività di 'sette microohm per metro' si scrive $7\,\mu\Omega\,\mathrm{m}$ oppure $7\,\mu\Omega\cdot\mathrm{m}$, ma sarebbe sbagliatissimo scriverla $7\,\mu\Omega/\mathrm{m}$ interpretando la preposizione 'per' all'inglese.[5]

[5] In italiano 'diviso' si dice 'al', non 'per'; al massimo si può dire 'per ogni'.

Capitolo 10

LaTeX: la matematica avanzata

Questo capitolo è dedicato a chi ha bisogno di scrivere espressioni matematiche un po' più complesse di quelle che si possono comporre con i comandi base di LaTeX esposti nel capitolo precedente.

È comodo fare uso del pacchetto di estensione **amsmath**. Questa estensione è il frutto della collaborazione con l'American Mathematical Society che, dai tempi del plain TeX originale, anzi del TeX 78, la primissima versione del sistema TeX, ha sempre collaborato ed è sempre stata strenua sostenitrice di questo software, ne ha sponsorizzato il 'trade mark', ed è stata la prima società scientifica ad accettare i contributi alle proprie riviste scritti in plain TeX, in AMSTeX e infine in AMSLaTeX; vale a dire con i programmi standard estesi con i pacchetti preparati dai programmatori della Società stessa e poi messi a disposizione di tutta la comunità degli utenti. La American Mathematical Society ha anche messo a disposizione i file di classe specializzati per scrivere articoli destinati alle sue varie pubblicazioni; ora questa pratica è seguita da tutte le più importanti riviste scientifiche internazionali; anche gli atti di molti congressi sono scritti usando direttamente i file sorgente degli autori delle memorie presentate a quei congressi.

Il pacchetto principe per l'utente generico è il pacchetto **amsmath**. Questo pacchetto mette a disposizione una miriade di comandi nuovi, da quelli che servono per usare altri simboli, oltre a quelli già molto numerosi presenti normalmente in LaTeX, a quelli che servono per predisporre incolonnamenti particolari, a quelli che servono per comporre segni o strutture matematiche avanzate.

10.1 I simboli di **amsmath**

Il pacchetto **amsmath** consente di accedere ai simboli delle tabelle 10.1 e 10.2.

Se insieme a **amsmath** si invocano anche **amsfonts** si hanno disponibili anche i segni del vecchio pacchetto **ltxsymb** il cui uso, come si è avuto occasione di commentare, è oggi sconsigliato. D'altra parte **amsfonts** mette a disposizione anche altri font, in particolare i font 'Euler frakture' che in matematica possono talvolta risultare più efficaci dei caratteri calligrafici.

\ulcorner	⌜	\urcorner	⌝	\llcorner	⌞
\lrcorner	⌟	\dashrightarrow	⇢	\dashleftarrow	⇠
\widehat	^	\widetilde	~	\dasharrow	⇢
\rightleftharpoons	⇌	\leftrightharpoons	⇋	\angle	∠
\hbar	ℏ	\sqsubset	⊏	\sqsupset	⊐
\mho	℧	\square	□	\lozenge	◊
\vartriangleright	▷	\vartriangleleft	◁	\trianglerighteq	⊵
\trianglelefteq	⊴	\rightsquigarrow	⇝	\boxdot	⊡
\boxplus	⊞	\boxtimes	⊠	\boxminus	⊟
\blacksquare	■	\centerdot	·	\blacklozenge	⧫
\circlearrowright	↻	\circlearrowleft	↺	\Vdash	⊩
\Vvdash	⊪	\vDash	⊨	\twoheadrightarrow	↠
\twoheadleftarrow	↞	\leftleftarrows	⇇	\rightrightarrows	⇉
\upuparrows	⇈	\downdownarrows	⇊	\upharpoonright	↾
\downharpoonright	⇂	\upharpoonleft	↿	\downharpoonleft	⇃
\rightarrowtail	↣	\leftarrowtail	↢	\leftrightarrows	⇆
\rightleftarrows	⇄	\Lsh	↰	\Rsh	↱
\rightsquigarrow	⇝	\leftrightsquigarrow	↭	\looparrowleft	↫
\looparrowright	↬	\circeq	≗	\succsim	≿
\gtrsim	≳	\gtrapprox	⪆	\multimap	⊸
\therefore	∴	\because	∵	\doteqdot	≑
\triangleq	≜	\precsim	≾	\lesssim	≲
\lessapprox	⪅	\eqslantless	⪕	\eqslantgtr	⪖
\curlyeqprec	⋞	\curlyeqsucc	⋟	\preccurlyeq	≼
\leqq	≦	\leqslant	⩽	\lessgtr	≶
\backprime	‵	\risingdotseq	≓	\fallingdotseq	≒
\succcurlyeq	≽	\geqq	≧	\geqslant	⩾
\gtrless	≷	\vartriangleright	▷	\vartriangleleft	◁
\trianglerighteq	⊵	\trianglelefteq	⊴	\bigstar	★
\between	≬	\blacktriangledown	▼	\blacktriangleright	▶
\blacktriangleleft	◀	\vartriangle	△	\blacktriangle	▲
\triangledown	▽	\eqcirc	≖	\lesseqgtr	⋚
\gtreqless	⋛	\lesseqqgtr	⪋	\gtreqqless	⪌
\Rrightarrow	⇛	\Lleftarrow	⇚	\veebar	⊻
\barwedge	⊼	\doublebarwedge	⩞	\measuredangle	∡
\sphericalangle	∢	\varpropto	∝	\smallsmile	⌣
\smallfrown	⌢	\Subset	⋐	\Supset	⋑
\Cup	⋓	\Cap	⋒	\curlywedge	⋏
\curlyvee	⋎	\leftthreetimes	⋋	\rightthreetimes	⋌
\subseteqq	⫅	\supseteqq	⫆	\bumpeq	≏
\Bumpeq	≎	\lll	⋘	\ggg	⋙
\circledS	Ⓢ	\pitchfork	⋔	\dotplus	∔
\backsim	∽	\backsimeq	⋍	\complement	∁
\intercal	⊺	\circledcirc	⊚	\circledast	⊛
\circleddash	⊝				

Tabella 10.1: Prima serie di simboli accessibili con il pacchetto `amsmath`

`\lvertneqq`	$\lvertneqq$	`\gvertneqq`	$\gvertneqq$	`\nleq`	$\nleq$
`\ngeq`	$\ngeq$	`\nless`	$\nless$	`\ngtr`	$\ngtr$
`\nprec`	$\nprec$	`\nsucc`	$\nsucc$	`\lneqq`	$\lneqq$
`\gneqq`	$\gneqq$	`\nleqslant`	$\nleqslant$	`\ngeqslant`	$\ngeqslant$
`\lneq`	$\lneq$	`\gneq`	$\gneq$	`\npreceq`	$\npreceq$
`\nsucceq`	$\nsucceq$	`\precnsim`	$\precnsim$	`\succnsim`	$\succnsim$
`\lnsim`	$\lnsim$	`\gnsim`	$\gnsim$	`\nleqq`	$\nleqq$
`\ngeqq`	$\ngeqq$	`\precneqq`	$\precneqq$	`\succneqq`	$\succneqq$
`\precnapprox`	$\precnapprox$	`\succnapprox`	$\succnapprox$	`\lnapprox`	$\lnapprox$
`\gnapprox`	$\gnapprox$	`\nsim`	$\nsim$	`\ncong`	$\ncong$
`\diagup`	$\diagup$	`\diagdown`	$\diagdown$	`\varsubsetneq`	$\varsubsetneq$
`\varsupsetneq`	$\varsupsetneq$	`\nsubseteqq`	$\nsubseteqq$	`\nsupseteqq`	$\nsupseteqq$
`\subsetneqq`	$\subsetneqq$	`\supsetneqq`	$\supsetneqq$	`\varsubsetneqq`	$\varsubsetneqq$
`\varsupsetneqq`	$\varsupsetneqq$	`\subsetneq`	$\subsetneq$	`\supsetneq`	$\supsetneq$
`\nsubseteq`	$\nsubseteq$	`\nsupseteq`	$\nsupseteq$	`\nparallel`	$\nparallel$
`\nmid`	$\nmid$	`\nshortmid`	$\nshortmid$	`\nshortparallel`	$\nshortparallel$
`\nvdash`	$\nvdash$	`\nVdash`	$\nVdash$	`\nvDash`	$\nvDash$
`\nVDash`	$\nVDash$	`\ntrianglerighteq`	$\ntrianglerighteq$	`\ntrianglelefteq`	$\ntrianglelefteq$
`\ntriangleleft`	$\ntriangleleft$	`\ntriangleright`	$\ntriangleright$	`\nleftarrow`	$\nleftarrow$
`\nrightarrow`	$\nrightarrow$	`\nLeftarrow`	$\nLeftarrow$	`\nRightarrow`	$\nRightarrow$
`\nLeftrightarrow`	$\nLeftrightarrow$	`\nleftrightarrow`	$\nleftrightarrow$	`\divideontimes`	$\divideontimes$
`\varnothing`	$\varnothing$	`\nexists`	$\nexists$	`\Finv`	$\Finv$
`\Game`	$\Game$	`\eth`	$\eth$	`\eqsim`	$\eqsim$
`\beth`	$\beth$	`\gimel`	$\gimel$	`\daleth`	$\daleth$
`\lessdot`	$\lessdot$	`\gtrdot`	$\gtrdot$	`\ltimes`	$\ltimes$
`\rtimes`	$\rtimes$	`\shortmid`	$\shortmid$	`\shortparallel`	$\shortparallel$
`\smallsetminus`	$\smallsetminus$	`\thicksim`	$\thicksim$	`\thickapprox`	$\thickapprox$
`\approxeq`	$\approxeq$	`\succapprox`	$\succapprox$	`\precapprox`	$\precapprox$
`\curvearrowleft`	$\curvearrowleft$	`\curvearrowright`	$\curvearrowright$	`\digamma`	$\digamma$
`\varkappa`	$\varkappa$	`\Bbbk`	$\Bbbk$	`\hslash`	$\hslash$
`\backepsilon`	$\backepsilon$				

Tabella 10.2: Seconda serie di simboli accessibili con il pacchetto `amsmath`

Invece, se non occorre tutto il macchinario di **amsmath** ma occorrono solo alcuni simboli, in particolare le lettere maiuscole ad aste raddoppiate, si può usare solo il pacchetto **amsfonts** che definisce il comando `\mathbb` per usarle. Qui, nelle tabelle citate, si è riportato tutto l'insieme dei simboli componibili per mezzo di tutti i pacchetti associati a **amsmath** in modo che il lettore abbia la consapevolezza della potenza del sistema TeX per quanto riguarda la simbologia disponibile.

10.2 Le estensioni dei font matematici

La AMS ha messo a disposizione due altre polizze di font chiamate criptica-mente `msam` e `msbm` i cui simboli sono direttamente accessibili con il pacchetto **amsmath**, oppure, se non si vuole richiamare l'intero pacchetto, con **amsfonts**.

10.3 I sistemi di equazioni

I sistemi di equazioni sono formati da un certo numero di equazioni incolonnate una sopra l'altra, di solito allineate in modo che i segni di uguaglianza (o qualunque altro segno di relazione) siano allineati verticalmente.

LaTeX è nato con un ambiente *eqnarray* (anche asteriscato) che consente di eseguire questo incolonnamento; per esempio:

$$\begin{aligned} 3x + 7y - 2z &= 13 && (10.1)\\ 2y + 3z &= 12 && (10.2)\\ 5x - 3y + 4z &= 6 && (10.3) \end{aligned}$$

in cui le equazioni sono numerate singolarmente da 10.1 a 10.3 e vi si può fare riferimento singolarmente usando il comando `\label` alla fine di ciascuna riga. Il sistema esposto è stato scritto con il codice

```
\begin{eqnarray}
  3x + 7y - 2z &=& 13 \label{equ:sistema1}\\
       2y + 3z &=& 12 \label{equ:sistema2}\\
  5x - 3y + 4z &=& 6  \label{equ:sistema3}
\end{eqnarray}
```

Se invece di usare l'ambiente *eqnarray* si fosse usato l'ambiente *eqnarray** le equazioni non sarebbero state numerate.

Purtroppo questo ambiente è nato con un 'peccato originale'; se il lettore osserva bene, constata che gli spazi a destra e a sinistra dei segni di uguaglianza nelle equazioni 10.1–10.3 è maggiore che non nelle equazioni del capitolo precedente. Non si sa se questo fosse voluto o se sia successo per caso, ma quando dal vecchio LaTeX 209 si è passati al nuovo LaTeX 2_ε, questo difetto è stato conservato, apparentemente per mantenere una compatibilità con i file predisposti per essere composti con la vecchia versione.

Oggi gli ambienti *eqnarray* e *eqnarray** non dovrebbero venire più usati, ma si dovrebbero usare solo gli ambienti predisposti dal pacchetto **amsmath**.

10.4 Gli ambienti di composizione di **amsmath**

Il pacchetto **amsmath** mette a disposizione una serie di ambienti per comporre e/o incolonnare equazioni o per raccoglierle in modo ordinato, ovvero per scriverle su più righe in modo professionale, anche se l'intervento dell'autore per conoscere il ritmo delle espressioni matematiche è fondamentale.

Gli ambienti disponibili sono raggruppati nella tabella 10.3.

equation	*equation**	*align*	*align**
gather	*gather**	*flalign*	*flalign**
multline	*multline**	*alignat*	*alignat**
aligned	*split*	*subequations*	

Tabella 10.3: Gli ambienti di allineamento di **amsmath**

Tutti gli ambienti hanno la versione asteriscata che differisce da quella non asteriscata per la mancanza delle numerazione dell'equazione; solo *split* e *aligned* ne sono privi perché devono essere usati all'interno degli altri ambienti, non possono essere usati da soli. *subequations* invece ne è privo perché serve per numerare le equazioni in un modo particolare.

La numerazione può anche essere esclusa inserendo il comando `\notag` prima del segno di 'a capo' di ogni riga dell'allineamento; alternativamente il comando `\tag` consente di inserire una etichetta a propria scelta come per esempio nell'equazione

$$a^2 + b^2 = c^2 \qquad \text{(Teorema di Pitagora)}$$

ottenuto scrivendo

```
\begin{equation}
a^2 +b^2 = c^2 \tag{Teorema di Pitagora}
\end{equation}
```

Tutti gli ambienti di incolonnamento usano il segno `&` per passare da una colonna all'altra e non vogliono mai due segni a cavallo degli operatori di relazione, come succede con *eqnarray*, ma ne vogliono uno solo *prima* dell'operatore.

10.4.1 L'ambiente **equation**

Di *equation* si è già detto quanto basta; il suo compagno asteriscato, a parte la numerazione, consente di contenere tutti gli altri ambienti che possono essere inclusi negli ambienti numerabili, in particolare *split* per scrivere lunghe equazioni su più righe.

10.4.2 L'ambiente **aligned**

L'ambiente *aligned* serve per comporre un allineamento di equazioni o di espressioni matematiche all'interno di un altro ambiente che sia in grado, eventualmente, di assegnare un numero a questo allineamento. Per esempio si può scrivere

$$\begin{aligned}\mathrm{Ker}_n(x) = &-[\log(x/2)+\gamma] + \tfrac{1}{4}\pi\,\mathrm{Bei}_n(x)\\ &+\frac{1}{2}\sum_{k=0}^{n-1}\frac{(n-k-1)!(x/2)^{2k-n}}{k!}\cos\frac{(3n+2k)\pi}{4}\\ &+\frac{1}{2}\sum_{k=0}^{\infty}\frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]\cos\frac{(3n+2k)\pi}{4}\end{aligned} \tag{10.4}$$

Questo ambiente produce risultati molto simili a quelli dell'ambiente *split* a cui si rinvia per i commenti.

10.4.3 L'ambiente split

L'ambiente *split* serve per dividere una lunga espressione matematica fra più righe.

L'equazione su più righe 10.4, resa con *aligned*, con *split* diventa:

$$\begin{aligned}\mathrm{Ker}_n(x) = &-[\log(x/2)+\gamma] + \tfrac{1}{4}\pi\,\mathrm{Bei}_n(x)\\ &+\frac{1}{2}\sum_{k=0}^{n-1}\frac{(n-k-1)!(x/2)^{2k-n}}{k!}\cos\frac{(3n+2k)\pi}{4}\\ &+\frac{1}{2}\sum_{k=0}^{\infty}\frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]\cos\frac{(3n+2k)\pi}{4}\end{aligned} \tag{10.5}$$

Il risultato è stato ottenuto con

```
\begin{equation}
  \begin{split}
    \Ker_n(x) & = -[\log(x/2)+\gamma]
                          +{\textstyle\frac{1}{4}}\pi\Bei_n(x)\\
              & \quad+\frac{1}{2}\sum_{k=0}^{n-1}
                       \frac{(n-k-1)!(x/2)^{2k-n}}{k!}
                                      \cos\frac{(3n+2k)\pi}{4}\\
              & \qquad+\frac{1}{2}\sum_{k=0}^\infty
                \frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]
                                        \cos\frac{(3n+2k)\pi}{4}
  \end{split}
  \label{equ:Ker2}
\end{equation}
```

La differenza fra *split* e *aligned* è che il primo ambiente, oltre a consentire allineamenti verticali come per altro fa anche il secondo ambiente, produce un risultato finale che, a parte il suo contenuto, ha esattamente la larghezza della riga in cui compare, mentre *aligned* produce il suo risultato finale sotto forma di un oggetto che è largo quanto il suo contenuto. In tal modo questo oggetto può costituire un elemento di costruzione di un oggetto più grande. Questi due ambienti, inoltre, differiscono dall'ambiente *multline* perché consentono degli allineamenti verticali, mentre *multline* non lo consente, come si vedrà tra poco.

10.4.4 L'ambiente multline

La stessa equazione su tre righe composta con l'ambiente *multline* diventa

$$\begin{aligned}\mathrm{Ker}_n(x) = &-[\log(x/2)+\gamma]+\tfrac{1}{4}\pi\,\mathrm{Bei}_n(x)\\ &+\frac{1}{2}\sum_{k=0}^{n-1}\frac{(n-k-1)!(x/2)^{2k-n}}{k!}\cos\frac{(3n+2k)\pi}{4}\\ &+\frac{1}{2}\sum_{k=0}^{\infty}\frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]\cos\frac{(3n+2k)\pi}{4}\end{aligned}\tag{10.6}$$

con il codice sorgente

```
\begin{multline}
  \Ker_n(x)  = -[\log(x/2)+\gamma]
                        +{\textstyle\frac{1}{4}}\pi\Bei_n(x)\\
  +\frac{1}{2}\sum_{k=0}^{n-1}
              \frac{(n-k-1)!(x/2)^{2k-n}}{k!}
                                    \cos\frac{(3n+2k)\pi}{4}\\
  +\frac{1}{2}\sum_{k=0}^\infty
       \frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]
                                      \cos\frac{(3n+2k)\pi}{4}
\end{multline}
```

Si noti in particolare la mancanza dei segni di incolonnamento `&` e la mancanza dei comandi di spaziatura `\quad` e `\qquad` presenti negli esempi precedenti; infatti `multline` compone la prima riga allineata con il margine sinistro delle equazioni; allinea l'ultima espressione con il margine destro tenendo conto del numero dell'equazione; infine allinea tutte le altre righe in modo da centrarle nello spazio disponibile. Se il numero dell'equazione è a destra, è in linea con l'ultima riga, oppure sotto all'ultima riga; se si decide di comporre con i numeri delle equazioni a sinistra, il numero è allineato con la prima riga oppure è collocato prima della prima riga; in entrambi i casi non c'è mai sovrapposizione fra il numero e le espressioni.

10.4.5 L'ambiente gather

L'ambiente *gather* serve per raccogliere ordinatamente da due a più equazioni una per riga, senza nessun incolonnamento particolare ma centrando le varie equazioni; in un certo senso poterebbe essere visto come una sequenza di ambienti *equation*; con questo ambiente si possono raccogliere le equazioni di Maxwell, per esempio, nella forma:

$$\nabla\times\vec{E} = -\frac{\partial\vec{B}}{\partial t}\tag{10.7}$$

$$\nabla\times\vec{H} = \frac{\partial\vec{D}}{\partial t}+\vec{J}\tag{10.8}$$

$$\nabla\cdot\vec{D} = \rho\tag{10.9}$$

$$\nabla\cdot\vec{B} = 0\tag{10.10}$$

ottenuta scrivendo

```
\begin{gather}
  \nabla\times\vec{E}=-\frac{\partial\vec{B}}{\partial t}\\
  \nabla\times\vec{H}=
                    \frac{\partial\vec{D}}{\partial t}+\vec{J}\\
  \nabla \cdot\vec{D}= \rho\\
  \nabla \cdot\vec{B}= 0
\end{gather}
```

10.4.6 L'ambiente **align**

Al contrario, usando l'ambiente *align* le stesse equazioni di Maxwell diventano:

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{10.11}$$

$$\nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} + \vec{J} \tag{10.12}$$

$$\nabla \cdot \vec{D} = \rho \tag{10.13}$$

$$\nabla \cdot \vec{B} = 0 \tag{10.14}$$

ottenute scrivendo:

```
\begin{align}
  \nabla\times\vec{E}&= -\frac{\partial\vec{B}}{\partial t}\\
  \nabla\times\vec{H}&=
                    \frac{\partial\vec{D}}{\partial t}+\vec{J}\\
  \nabla \cdot\vec{D}&= \rho\\
  \nabla \cdot\vec{B}&= 0
\end{align}
```

Si nota che l'unica differenza consiste nel comando di incolonnamento `&` inserito prima di ciascun segno di uguaglianza.

All'interno di un ambiente *align* possono comparire diversi incolonnamenti, non uno solo come nell'esempio precedente; per esempio le equazioni di Maxwell potrebbero venire riorganizzate così:

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \qquad \nabla \cdot \vec{D} = \rho \tag{10.15}$$

$$\nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} + \vec{J} \qquad \nabla \cdot \vec{B} = 0 \tag{10.16}$$

usando la scrittura:

```
\begin{align}
  \nabla\times\vec{E}&=-\frac{\partial\vec{B}}{\partial t}
                    &\nabla \cdot\vec{D}&=\rho\\
  \nabla\times\vec{H}&=\frac{\partial\vec{D}}{\partial t}+\vec{J}
                    &\nabla \cdot\vec{B}&=0
\end{align}
```

10.4.7 L'ambiente flalign

L'ambiente *flalign* differisce dall'ambiente *align* solo per il fatto che le espressioni allineate vengono spostate al massimo all'esterno della riga dove compaiono.

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \qquad \nabla \cdot \vec{D} = \rho \quad (10.17)$$

$$\nabla \times \vec{H} = \vec{J} + \frac{\partial \vec{D}}{\partial t} \qquad \nabla \cdot \vec{B} = 0 \quad (10.18)$$

e si vede chiaramente che questo ambiente va meglio quando le equazioni non sono numerate e quando le espressioni sono sufficientemente lunghe; nelle altre circostanze è meglio usare `align`

10.4.8 L'ambiente alignat

L'ambiente *alignat* è un po' particolare, perché accetta un numero come argomento del comando di apertura che specifica quanti punti di allineamento compaiono in ogni colonna; la regola dice: conta il massimo numero di segni `&` che compaiono in ogni riga, aggiungi 1 e dividi per 2. Così l'esempio del manuale diventa:

$$\begin{aligned} x &= y_1 - y_2 + y_3 - y_5 + y_8 - \dots && \text{by eq. (3.21)} && (10.19)\\ &= y' \circ y^* && \text{by eq. (4.1)} && (10.20)\\ &= y(0)y' && \text{by Axiom 1} && (10.21) \end{aligned}$$

prodotto con il codice

```
\begin{alignat}{2}
  x&= y_1 - y_2 + y_3 -y_5 +y_8 -\dots
                        &\quad & \text{by eq.~(3.21)}\\
   &=y'\circ y^*        &      & \text{by eq.~(4.1)}\\
   &= y(0)y'            &      & \text{by Axiom 1}
\end{alignat}
```

e nel quale si osserva la presenza del comando `\text`; questo comando serve per scrivere del testo all'interno di espressioni matematiche; se questo testo contiene una espressione matematica in linea, questa va esplicitamente racchiusa fra i delimitatori `\(` e `\)`.

10.4.9 L'ambiente subequations

L'ambiente *subequations* è particolare perché le equazioni all'interno di questo ambiente vengono numerate con un contatore subalterno così da produrre numeri del tipo (10.22a), (10.22b) eccetera. Per esempio:

$$\mathbf{Im}[F(\sigma + \mathrm{i}\,0)] = 0 \qquad \forall \sigma > 0 \qquad (10.22a)$$

$$F(p) \neq \infty \qquad \forall \sigma > 0 \qquad (10.22b)$$

$$\lim_{p \to \mathrm{i}\,\omega_0} (p - \mathrm{i}\,\omega_0) F(p) = k \qquad \forall \omega_0 \text{ e con } 0 \leq k < \infty \qquad (10.22c)$$

$$\lim_{p \to \infty} F(p)/p = h \qquad \text{con } 0 \leq h < \infty \qquad (10.22d)$$

$$\mathbf{Re}[F(0 + \mathrm{i}\,\omega)] \geq 0 \qquad \forall \omega \qquad (10.22e)$$

ottenuto con il codice seguente:

```
\DeclareMathOperator{\uimm}{\mathrm{i}}          % nel preambolo
...
\begin{subequations}
                                                    \label{equ:sube}
  \begin {align}
     \Im[F(\sigma+\uimm 0)]  &= 0     &\forall\sigma>0\\
     F(p)                    &\not=\infty&\forall\sigma>0\\
     \lim_{p\to\uimm\omega_0}(p-\uimm\omega_0)F(p)&=k
                             & \forall\omega_0\text{ e con }
                                          0\le k<\infty\\
     \lim_{p\to\infty} F(p)/p&=h      &\text{con }0\le h<\infty\\
     \Re[F(0+\uimm\omega)]   &\ge 0   &\forall\omega
                                          \label{equ:sube5}
 \end{align}
\end{subequations}
```

Con il comando `\label` inserito prima dell'ambiente *align*, ma dentro all'ambiente *subequations* si recupera il numero dell'intero sistema di condizioni 10.22; invece con il comando `\label` inserito all'interno dell'ambiente *align* si recupera il 'numero' assegnato a ciascuna condizione, per esempio all'ultima condizione 10.22e si fa riferimento mediante `\ref{equ:sube5}`.

La definizione dell'unità immaginaria`\uimm` è un po' particolare nel senso che essa è definita come un operatore, invece che come una semplice variabile. Si noti che se si usasse il comando primitivo `\mathop`, la definizione dovrebbe contenere oltre alla 'i' in tondo anche uno strut o dello spazio matematico al fine di centrare la 'i' rispetto al suo asse matematico. È questa una ragione in più per ricorrere ai comandi di definizione disponibili con il pacchetto **amsmath**, che permettono di concentrarsi sul significato intrinseco di ciò che si sta definendo, senza badare ai dettagli compositivi a livello di base.

Esercizio 10.1 Il lettore provi a ripetere la composizione di questo sistema di condizioni 10.22 usando il comando `\mathop` ma omettendo lo 'strut' matematico nella definizione di `\uimm`. Che cosa succede?

10.5 Altri comandi e ambienti

Il pacchetto **amsmath** offre numerosi altri comandi e ambienti che consentono di comporre la matematica in modo estremamente professionale.

10.5.1 Definizione di operatori funzionali

Qui si cita nuovamente il comando per definire i nomi di nuovi operatori funzionali:

`\DeclareMathOperator{`⟨*operatore*⟩`}{`⟨*definizione*⟩`}`

così che se si volesse definire l'operatore per l'arcoseno iperbolico 'arsinh' (che non è definito né da LaTeX standard, né da `amsmath`) basterebbe dichiarare nel preambolo:

```
\DeclareMathOperator{\arsinh}{\mathrm{arsinh}}
```

Il comando asteriscato serve per definire operatori funzionali con i limiti.

10.5.2 Le frazioni in generale

Il pacchetto **amsmath** consente di scrivere le frazioni e gli altri costrutti simili alle frazioni con tre comandi distinti: il nome 'normale' che compone in modo diverso in modo testo rispetto al modo display, il nome testuale per comporre sempre e comunque in modo testo, e il modo 'enfatico' per scrivere sempre e comunque in modo display; per esempio le frazioni normalmente vengono composte con il comando `\frac` ma il modo testuale viene composto con `\tfrac` e il modo enfatico con `\dfrac`. L'operatore binomiale, per esempio si compone con `\binom`:

$$\binom{n}{k} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{1\cdot 2\cdot 3\cdots k}$$

ma se proprio si volesse indicare il coefficiente binomiale in modo display mentre si è in modo testo, si userebbe `\dbinom` e si otterrebbe $\dbinom{n}{k}$; come si vede, questo è un tipo di scrittura da evitare assolutamente quando si compone matematica in linea con il testo, perché altera vistosamente l'avanzamento di riga.

10.5.3 Le frazioni continue

Fra i comandi utilissimi conviene citare quello per la composizione delle frazioni continue `\cfrac` con il quale si possono comporre facilmente espressioni del tipo

$$\tanh z = \cfrac{z}{1+\cfrac{z^2}{3+\cfrac{z^2}{5+\cfrac{z^2}{7+\cfrac{z^2}{\cdots}}}}} \tag{10.23}$$

composta con i comandi

```
\begin{equation}
  \tanh z = \cfrac{z}{1+\cfrac{z^2}{3+\cfrac{z^2}{5+
                      \cfrac{z^2}{7+\cfrac{z^2}{\cdots}}}}}
\end{equation}
```

10.5.4 Il testo intercalato alle equazioni

Un altro comando molto interessante è `\intertext`; esso consente di inserire un (breve) testo che interrompe un incolonnamento di equazioni senza perdere i parametri dell'incolonnamento, così che le successive equazioni mantengono lo stesso incolonnamento; per esempio

$$0 = a_4x^4 + a_2x^2 + a_0 \tag{10.24}$$

diventa una semplice equazione di secondo grado

$$0 = a_4 t^2 + a_2 t + a_0 \tag{10.25}$$

se si pone

$$t = x^2 \tag{10.26}$$

Simile a `\intertext` è il comando `\text`, già visto; esso serve per inserire un po' di testo dentro una formula, specialmente negli enunciati delle condizioni, ma non solo; esso serve anche e tipicamente all'interno dell'ambiente *cases*; l'equazione 9.5 usata per mostrare la composizione dei delimitatori estensibili e del delimitatore invisibile, può essere scritta mediante questo ambiente e questo comando nella maniera seguente:

$$x_{1,2} = \begin{cases} \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a} & \text{se } b^2 - 4ac > 0 \\ -\dfrac{b}{2a} & \text{se } b^2 - 4ac = 0 \\ \dfrac{-b \pm \mathrm{i}\sqrt{4ac - b^2}}{2a} & \text{se } b^2 - 4ac < 0 \end{cases} \tag{10.27}$$

scrivendo nel file sorgente

```
\begin{equation}
x_{1,2}=
\begin{cases}
  \dfrac{-b\pm\sqrt{b^2-4ac}}{2a}
                          &\text{se \(b^2-4ac>0\)}\\[1.5ex]
  -\dfrac{b}{2a}          &\text{se \(b^2-4ac=0\)}\\[1.5ex]
  \dfrac{-b\pm\uimm\sqrt{4ac-b^2}}{2a}
                          &\text{se \(b^2-4ac<0\)}
\end{cases}
\end{equation}
```

Si noti che nell'ambito dell'argomento di `\text` la condizione è nuovamente composta come matematica in linea, quindi è necessario usare gli appositi comandi `\(` e `\)` come si fa normalmente quando si compone all'interno dei capoversi. Si noti anche l'uso del comando `\dfrac` per obbligare LaTeX a comporre le frazioni in modo display, nonostante spontaneamente esso comporrebbe quelle frazioni in modo testo; l'ulteriore spaziatura mediante l'argomento facoltativo `\\[1.5ex]` serve per compensare l'effetto prodotto dall'uso delle frazioni in display.

Esercizio 10.2 Che cosa succederebbe se nell'equazione 10.27 non si usasse `\dfrac`?

Esercizio 10.3 Che cosa succederebbe nell'equazione 10.27 se si usasse `\\` senza il suo argomento facoltativo?

10.5.5 Le frecce estensibili

Le frecce estensibili diventano lunghe quanto i loro argomenti; si tratta dei comandi `\xleftarrow` e `\xrightarrow` che accettano due argomenti, uno obbligatorio che sarà collocato sopra la freccia e uno facoltativo che verrà collocato sotto la freccia:

$$A \xleftarrow{n+\mu-1} B \xrightarrow[T]{n\pm i-1} C \tag{10.28}$$

composto con

```
A \xleftarrow{n+\mu-1} B \xrightarrow[T]{n\pm i-1} C
```

10.5.6 Gli indici incolonnati

Talvolta può essere necessario indicare gli indici di somma o di prodotto con diverse indicazioni o condizioni, come in

$$\sum_{\substack{0\le i\le m\\ 0<j<n}} P(i,j) \tag{10.29}$$

Questo risultato si ottiene mediante `\substack` come si vede dall'esempio composto con

```
\begin{equation}
\sum_{\substack{0\le i\le m\\
                0<j<n
               }
     } P(i, j)
\end{equation}
```

10.5.7 Gli integrali multipli

Nel caso di integrali multipli LaTeX senza estensioni necessiterebbe di un aggiustamento della spaziatura; si osservi

$$\int\int\int\int \dots \mathrm{d}w\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z \qquad \iiiint \dots \mathrm{d}w\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z \tag{10.30}$$

Nel primo caso si è usato quattro volte il segno di integrale, mentre nel secondo caso si è usato il comando per l'integrale quadruplo disponibile con il pacchetto **amsmath** che mette a disposizione anche i comandi per l'integrale doppio e per quello triplo; in totale i comandi disponibili sono `\int` per l'integrale semplice, `\iint` per l'integrale doppio, `\iiint` per l'integrale triplo e `\iiiint` per l'integrale quadruplo; infine c'è anche `\idotsint` per l'integrale multiplo.

10.5.8 L'operatore differenziale

Si noti nell'equazione 10.30 il segno di differenziale usato come un operatore speciale che lascia il suo normale spazio alla sua sinistra ma non lascia spazio alla sua destra; inoltre esso è scritto in tondo come prescrivono le norme ISO. È chiaro che non si tratta di un normale operatore definibile con il comando già descritto `\DeclareMathOperator`, perché con questa definizione il differenziale avrebbe lo spazio anche alla sua destra. Nel preambolo è stato definito l'operatore di differenziale `\diff` servendosi invece di un operatore 'vuoto', così:

```
\newcommand*{\diff}{\mathop{}\!\mathrm{d}}
```

I dettagli di `\newcommand` verranno visti nel capitolo 16; qui limitiamo la nostra attenzione sull'operatore `\mathop` il cui argomento viene definito come operatore matematico, salvo che il suo argomento è vuoto; nonostante ciò gli spazi che attorniano gli operatori sono rispettati, anche a destra; ma questo spazio a destra non ci vuole, quindi con il comando `\!` si arretra della stessa quantità, dopo di che si inserisce il segno del differenziale in tondo. Ecco, questa è una delle rare volte in cui si è usato un comando di spaziatura matematica (arretramento di uno spazio fine), ma lo si è fatto a ragion veduta per definire un comando per un operatore speciale come il differenziale.

Vale la pena di sottolineare che gli spazi del differenziale sono necessari per scrivere gli integrali, mentre servono di meno per scrivere le equazioni differenziali, ma ci pensa LaTeX a tenerli nella debita considerazione, come nella seguente equazione

$$a\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + b\frac{\mathrm{d}x}{\mathrm{d}t} + c = f(t)$$

10.5.9 I simboli corsivi matematici in nero

Una cosa che capita spesso in matematica è quella di usare il carattere nero all'interno di una formula composta con carattere medio. LaTeX senza estensioni non consente di scegliere altro che una formula tutta in carattere medio oppure tutta in carattere nero, non consente di mescolare questi due casi, a meno di non fare acrobazie di tipo prescrittivo, piuttosto che di mark-up. Inoltre se si desidera comporre una formula tutta nera bisogna dichiaralo prima di entrare in modo matematico, e di conseguenza occorre proteggere la formula 'nera' dentro un gruppo, oppure è necessario cancellare l'effetto di annerimento mediante l'apposito comando:

```
\boldmath
⟨ambiente matematico nero⟩
\unboldmath
```

All'interno di una espressione è possibile usare il carattere nero, ma solo per il tondo, non per il corsivo matematico. Il comando sarebbe `\mathbf` del tutto analogo al corrispondente comando per il testo `\textbf`; ma per il corsivo matematico non c'è nulla.

L'estensione fornita da **amsmath** consente di usare il comando `\boldsymbol` il cui argomento può essere qualunque simbolo letterale o operazionale matematico:

```
\boldsymbol{⟨simbolo matematico⟩}
```

a cui si aggiunge una scorciatoia (anche qualitativa) del *poor man bold*, `\pmb`, che simula il font nero replicando un segno chiaro tre volte con un leggero spostamento a destra e in alto (e questo rende meno definito il segno; ecco perché qualitativamente si tratta di una soluzione da pover uomo). La sintassi è la stessa:

```
\pmb{⟨simbolo matematico⟩}
```

In generale si sconsiglia di usare `\pbm`; con i font di serie con LaTeX non dovrebbe essere necessario usarlo altro che per i grandi operatori e per i delimitatori estensibili, ma non sembra una operazione molto frequente da fare.

Con questi comandi è possibile, per esempio, usare il corsivo matematico per indicare i vettori:

$$\boldsymbol{a} = a_x\boldsymbol{i} + a_y\boldsymbol{j} + a_z\boldsymbol{k}$$

10.5.10 Le espressioni matematiche riquadrate

Secondo lo scrivente per mettere in risalto una espressione matematica si può usare il meccanismo fornito da LaTeX, ma talvolta potrebbe essere preferibile racchiudere l'espressione dentro ad una cornice; **amsmath** mette a disposizione il comando `\boxed` come nell'esempio seguente

$$\boxed{x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}}$$

composto con

```
\[
\boxed{x_{1,2}= \frac{-b\pm\sqrt{b^2-4ac}}{2a}}
\]
```

10.6 Le matrici e i determinanti

Le matrici e i determinanti sono particolari tabelle che contengono solo, o prevalentemente, espressioni matematiche; **amsmath** mette a disposizione un certo numero di ambienti per comporre matrici variamente delimitate e perciò con significati matematici diversi.

L'ambiente base è ovviamente *matrix* e produce una matrice avente al massimo 10 colonne, ma sprovvista di delimitatori; anche gli altri ambienti consentono un massimo di 10 colonne, ma questo numero può venire cambiato se si imposta il contatore `MaxMatrixCols` con un valore diverso

`\setcounter{MaxMatrixCols}{`⟨*numero delle colonne*⟩`}`

Gli altri ambienti per costruire matrici o determinanti sono i seguenti: *pmatrix*, *bmatrix*, *Bmatrix*, *vmatrix* e *Vmatrix*; tutti questi racchiudono la matrice vera e propria all'interno di una coppia di delimitatori estensibili; a seconda del delimitatore la matrice ha un significato diverso. Gli ambienti *pmatrix*, *bmatrix* e *Bmatrix* racchiudono la matrice all'interno di una coppia di parentesi tonde, oppure quadre oppure graffe; l'ambiente *vmatrix* serve per comporre un determinante racchiudendo la matrice fra due aste verticali, mentre l'ambiente *Vmatrix* serve per comporre una norma racchiudendo la matrice fra due coppie di barre verticali.

Grazie a questi ambienti non è difficile comporre una espressione del tipo:

$$\begin{vmatrix} 1 & 1 & 0 & 0 \\ -1 & 2 & 1 & 0 \\ 0 & -1 & 3 & 1 \\ 0 & 0 & -1 & 4 \end{vmatrix} = 30 \tag{10.31}$$

mediante il codice

```
\begin{equation}
    \begin{vmatrix}
      1 & 1 & 0  & 0 \\
     -1 & 2 & 1  & 0 \\
      0 &-1 & 3  & 1 \\
      0 & 0 &-1  & 4
    \end{vmatrix}
  = 30
\label{equ:determinante}
\end{equation}
```

La scrittura nel file sorgente del codice per la composizione della matematica è più facilmente leggibile se si usa una scrittura strutturata; ci si rende più facilmente conto se ogni segno di apertura è in corrispondenza con il suo compagno di chiusura; usando questa pratica, nei limiti del possibile si evitano moltissimi errori e si compone più speditamente.

Una equazione matriciale diventa abbastanza semplice da comporre, specialmente se si ha l'accortezza di scrivere in modo strutturato nel file sorgente tutto quello che va ordinatamente inserito nelle matrici

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

Il codice sorgente è

```
\[
  \begin{pmatrix}
    a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\
    a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\
    a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\
    a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4}
  \end{pmatrix}
  \begin{pmatrix}
    x_1\\x_2\\x_3\\x_4
  \end{pmatrix}
  =
  \begin{pmatrix}
    y_1\\y_2\\y_3\\y_4
  \end{pmatrix}
\]
```

10.7 I diagrammi commutativi

I diagrammi commutativi sono particolari espressioni matematiche a metà strada fra un insieme di espressioni matematiche e un disegno che le mette in relazione le une con le altre.

Il pacchetto **amscd** serve per estendere **amsmath** definendo un ambiente *CD* con il quale è possibile disegnare diagrammi commutativi piani e senza frecce

diagonali. Per diagrammi più complessi il lettore deve rivolgersi a pacchetti di estensione più specializzati come **xypic** oppure **kuvio**.

Il diagramma commutativo

$$\begin{array}{ccc} S^{\mathcal{W}_\Lambda} \otimes T & \xrightarrow{j} & T \\ \downarrow & & \downarrow{\scriptstyle \mathrm{End}\ P} \\ (S \otimes T)/I & = & (Z \otimes T)/J \end{array}$$

è stato composto con il codice

```
\[
  \begin{CD}
    S^{\mathcal{W}_\Lambda}\otimes T @>j>> T \\
    @VVV                             @VV{\mathrm{End}\:P}V\\
    (S\otimes T)/I                   @=(Z\otimes T)/J
  \end{CD}
\]
```

Oltre ai vari comandi per inserire le frecce o le linee di collegamento fra le varie espressioni, si noti l'uso di `\mathcal` per scrivere caratteri 'calligrafici' e l'uso del comando di spaziatura `\:` per inserire uno spazio circa pari a quello interparola; il comando `\:` in matematica si comporta diversamente dai comandi analoghi per il modo testo `\enspace` o `\thinspace`, perché esso cambia valore a seconda che l'espressione matematica che si sta componendo sia in modo testo, in modo display, in modo degli indici del primo ordine o di quelli del secondo ordine.

Si prenda nota anche del fatto che l'ambiente *CD* deve essere realizzato all'interno di un ambiente matematico come quello prodotto da `\[...\]` oppure uno dei vari altri ambienti visti in questo capitolo.

10.8 La punteggiatura in matematica

La punteggiatura in matematica è tipograficamente un problema aperto perché non esistono norme che regolino la questione; è più una questione di consuetudini tipografiche ed editoriali. Anzi, ogni casa editrice si premura spesso di fornire ai suoi potenziali autori una breve guida in cui sono indicate le tradizioni compositive della casa editrice. Gli argomenti spaziano tutto lo scibile che può essere scritto, e vanno dai titoli correnti alle testatine e ai piedini, fino alle virgole del testo e la punteggiatura in matematica; naturalmente se la casa editrice riceve un contributo da pubblicare in forma direttamente stampabile, generalmente fornisce i fogli di stile nel linguaggio di video composizione che l'autore userà. Non sono rari i casi di case editrici che forniscono direttamente i file di classe e le eventuali estensioni per i libri da pubblicare dopo la composizione con LaTeX.

Tuttavia per quel che riguarda la punteggiatura della matematica qualche volta le informazioni o le prescrizioni sono lacunose, quindi l'autore deve supplire con la sua esperienza e la sua sensibilità.

Non dovrebbero sussistere problemi per le espressioni matematiche (brevi) che compaiono in linea con il testo, dove la punteggiatura dominante è quella testuale.

Direi che non dovrebbero esserci problemi nemmeno per la punteggiatura *interna* a ogni espressione matematica; questa punteggiatura specifica interna alle espressioni matematiche ha un suo particolare significato matematico che fa parte della sua 'grammatica'. I problemi maggiori nascono per la punteggiatura *esterna* alle espressioni matematiche in display.

Tuttavia cominciamo con i problemi minori di punteggiatura interna. In matematica si usano essenzialmente gli stessi segni di interpunzione che si usano nel testo, eccetto il punto interrogativo, salvo che alcuni di questi segni hanno un significato particolare; per esempio il punto esclamativo che segue un numero intero o un simbolo che rappresenta un numero intero, significa che di quel numero si considera il 'fattoriale':

$$n! = n(n-1)(n-2)\cdots 2\cdot 1$$

Lo stesso punto esclamativo può servire per cambiare il significato di un 'quantificatore' logico; per esempio $\exists!$ da taluni è usato nel senso di 'esiste ed è unico'.

I due punti hanno un significato particolare in espressioni logiche, così come (in Italia) possono indicare l'operazione di divisione, quindi non vengono usati come segno di interpunzione.

Il punto e virgola ha il solo significato di segno di interpunzione, mentre la virgola e il punto sono 'anfoteri', possono servire ora per un significato ora per un altro.

Scrivendo in inglese, il punto serve come separatore decimale e come punto fermo, mentre la virgola è un semplice segno di interpunzione, usato, come punteggiatura interna, per separare gli elementi di una lista.

Al contrario, in qualunque lingua diversa dall'inglese le norme ISO prescrivono la virgola nella funzione di separatore decimale, e quindi la virgola viene ad assumere due ruoli distinti; il punto, per altro sembrerebbe declassato a svolgere la sola funzione di segno di interpunzione (raro nell'interpunzione interna), ma poi rientra come separatore decimale nelle costanti numeriche 'reali' di tutti i linguaggi di programmazione. Nel capitolo 16 si vedrà come fare per convincere LaTeX a trattare correttamente le due funzioni della virgola in matematica.

La punteggiatura esterna è quella che pone i principali problemi. Mi spiego meglio: per *punteggiatura esterna* intendo sia la punteggiatura che compare in una espressione che contenga al suo interno anche delle parti testuali, sia la punteggiatura finale.

Non esistono regole 'grammaticali' codificate per gestire la punteggiatura esterna. Scrivendo per una data casa editrice, come già detto, è opportuno riferirsi alle sue regole editoriali, generalmente distribuite agli autori sotto forma di un opuscoletto dove sono descritte tutte le minuzie relative alla composizione, all'uso dei font, alla punteggiatura sia nel testo sia in matematica, alle abbreviazioni, alle maiuscole, agli accenti, alle 'd' eufoniche[1], alle 'i' prostetiche[2]. Scrivendo per proprio conto e, specialmente, quando si è editori di se stessi,

[1] La 'd' eufonica si pronuncia molto spesso quanto le congiunzioni 'e' e 'o' precedono parole che iniziano per vocale; spesso nello scrivere questa 'd' viene omessa come si è fatto in questa nota.

[2] La 'i' prostetica è quella che si pronuncia e talvolta si scrive aggiungendola all'inizio delle parole che cominciano con una 's' impura, quando la parola è preceduta dalle preposizioni 'in' e 'per'; per esempio: 'in iscuola', 'per istrada'.

bisogna decidere autonomamente e scegliere uno stile di punteggiatura esterna per la matematica.

Sul forum del GUIT alcuni argomenti sono stati dibattuti ampiamente e uno di questi argomenti trattava appunto della punteggiatura esterna. Sono emerse due scuole di pensiero fra loro antitetiche, ma ciascuna con pregi e difetti; l'unica cosa sulla quale le due scuole di pensiero concordano è che scelto uno stile di punteggiatura, quello deve essere usato costantemente e coerentemente; il lettore non deve essere confuso da un uso irregolare e non uniforme della punteggiatura. Le due scuole di pensiero dicono sostanzialmente quanto segue.

La punteggiatura esterna è necessaria in matematica come lo è per il testo Secondo questo orientamento la punteggiatura separa le varie parti di una espressione che contiene anche dei frammenti testuali e va inserita anche alla fine delle espressioni in display, sia sotto forma di virgole, di punti e virgola o di punti finali; in altre parole l'espressione matematica, anche se è in display, fa parte del testo e la punteggiatura ne esprime il ritmo esattamente come lo fa quando si compone del testo non matematico. Bisogna solo stare attenti a che i segni di punteggiatura non interferiscano con la parte matematica in senso stretto al punto da renderne difficile la lettura o, peggio ancora, al punto di indurre il lettore a farne una lettura ambigua o errata.

Siccome la punteggiatura non va staccata da quanto viene prima (a differenza del francese, nel quale, almeno il punto e virgola va spaziato con uno spazio di almeno un terzo di em da quanto viene prima), talvolta in italiano si potrebbro spaziare virgole e punti, suscettibili di introdurre ambiguità nella lettura delle espressioni, mediante uno spazio fine, che in matematica si rende semplicemente con il comando `\,`; si confronti, per esempio,

$$w = \frac{az+b}{cz+d}, \text{ con } a,b,c,d \in \mathbb{R};\ z,w \in \mathbb{C}$$

con

$$w = \frac{az+b}{cz+d}\,, \text{ con } a,b,c,d \in \mathbb{R};\ z,w \in \mathbb{C}.$$

La matematica si scrive senza punteggiatura esterna Secondo[3] questo orientamento la punteggiatura esterna nelle espressioni matematiche viene sostituita con degli spazi o degli incolonnamenti; se si deve separare un frammento testuale all'interno di una espressione, lo si fa usando spazi di almeno un quadrato, ma meglio ancora di un quadratone, rispettivamente con i comandi `\quad` e `\qquad`; un incolonnamento eseguito per esempio con l'ambiente *align* permette di separare le colonne delle condizioni dalle colonne contenenti espressioni matematiche, come si è fatto, per esempio, con il sistema di condizioni 10.22. Proseguendo l'esempio precedente si esamini ancora

$$w = \frac{az+b}{cz+d} \qquad \text{con } a,b,c,d \in \mathbb{R} \quad z,w \in \mathbb{C}$$

Secondo questa scuola di pensiero la punteggiatura finale è ridondante, perché gli spazi prima e dopo una formula in display e il modo di scrivere la prima

[3] Anche LaTeX toglie la punteggiatura finale ai titolini dei capoversi (composti con il comando `\paragraph`, come in questo caso) sostituendo questa punteggiatura con uno spazio interparola maggiore. La scuola di pensiero che non usa la punteggiatura esterna della matematica, è in linea con la pratica tipografica di usare spazi adeguati al posto della punteggiatura.

riga di testo che segue tale formula sono sufficienti a capire se la formula stessa chiude un capoverso, o se il periodo di cui fa parte finisce con la formula, oppure se continua dopo la formula; infatti se questa prima riga comincia con una lettera minuscola (e, ovviamente, non è rientrata) significa che il periodo continua dopo la formula; se questa prima riga non è rientrata ma comincia con una lettera maiuscola, la formula terminava il periodo, e ne comincia uno nuovo subito dopo, ma il capoverso non è terminato; se invece questa prima riga comincia con lettera maiuscola ed è rientrata, vuol dire che dopo la formula, che chiude il periodo, comincia un nuovo capoverso. Quello che l'assenza di punteggiatura non permette di capire è il quarto caso, cioè se la formula termina una frase o se la frase continua dopo la formula. In questi quattro casi l'altra scuola di pensiero avrebbe usato rispettivamente la virgola, il punto fermo, il punto e a capo (di fatto il punto alla fine della formula) oppure il punto e virgola.

Per altro, con riferimento all'espressione 10.27, dove si sarebbe messa la virgola di chiusura dell'espressione? In casi analoghi si è vista la virgola (o altri segni di interpunzione) allineata con l'asse matematico[4] dell'espressione, oppure a chiusura dell'ultima riga, ma non è semplice prendere una decisione compositiva in situazioni del genere.

Uso della punteggiatura e degli spazi in matematica Naturalmente nessuno vieta di usare sia la punteggiatura sia gli spazi; l'esempio riportato sopra diventerebbe

$$w = \frac{az + b}{cz + d}\,, \qquad \text{con } a, b, c, d \in \mathbb{R}; \quad z, w \in \mathbb{C}.$$

Come il lettore avrà notato, in questo testo si segue la seconda linea di pensiero; è una scelta del coordinatore, il quale si prende la responsabilità di questa scelta, ed è perfettamente conscio che i collaboratori seguaci dell'altra scuola di pensiero ne sono un po' delusi. L'omogeneità dello stile e, quindi, la coerenza compositiva, richiedono di non cambiare la punteggiatura da un paragrafo all'altro, a seconda di chi l'ha scritto. Il lettore invece, quando compone i suoi testi, seguirà le sue convinzioni e sceglierà il tipo di punteggiatura che preferisce.

10.9 Conclusioni

Non è questo il luogo per ripetere quanto è reperibile sulla documentazione del pacchetto **amsmath**; né è questo il luogo per illustrare altri pacchetti e altre funzionalità che la fantasia degli utenti di LaTeX è riuscita a inventare.

Il lettore sappia però che la storia non è finita qui; egli è invitato a esplorare gli archivi internazionali CTAN e scoprirà una varietà infinita di estensioni per la composizione della matematica che non potrà fare a meno di stupirlo; sappia però discernere quanto è stato escogitato alla buona per risolvere un problema contingente, da quanto è stato programmato professionalmente per uso generale.

[4] L'asse matematico è la linea orizzontale (invisibile) che passa a metà fra le due lineette che formano il segno =; gli operatori, le linee di frazione e tanti altri elementi delle espressioni matematiche dovrebbero essere spazialmente simmetrici rispetto a questo asse.

Capitolo 11

LaTeX: i caratteri da stampa

11.1 Introduzione

I caratteri da stampa ci sono familiari perché abbiamo una grande abitudine a leggere testi scritti a stampa; tuttavia siamo assolutamente ignoranti per quanto riguarda la loro storia, l'evoluzione, gli usi, i periodi storici, e quant'altro riguardi questi piccoli oggetti e i corrispondenti piccoli segni.

Tuttavia si sono già dette alcune cose in merito ai caratteri, che qui brevemente si richiamano.

11.2 Terminologia relativa ai caratteri

font è praticamente sinonimo di 'carattere' con l'unica avvertenza che in italiano si usa la parola 'carattere', specialmente al plurale, per designare una intera polizza di segni accomunati da una omogeneità stilistica, ma si usa anche, specialmente al singolare, per designare il singolo segno. Font è equivalente al primo significato, quello di 'polizza', mentre il singolo segno in inglese viene designato *glyph*; anche in italiano si potrebbe dire glifo, ma è assai raro vedere usata questa parola.

polizza indicava l'intera cassa di caratteri accomunati dalle stesse caratteristiche grafiche e tipografiche, incluso il numero esatto dei caratteri metallici presenti in ogni scomparto della cassa. Oggi che i caratteri metallici si usano assai poco e solo per lavori molto particolari, non ha più significato usare la parola 'polizza' per designarne anche la consistenza, in quanto questa è illimitata con i caratteri elettronici, ma la parola viene usata solo per indicare una collezione di segni; quasi sempre viene sostituita dall'equivalente parola inglese *font*, al punto che questa parola si può considerare definitivamente entrata nell'uso della lingua italiana.

corpo è la dimensione reale per i caratteri metallici, e quella virtuale per i caratteri elettronici, dell'altezza del rettangolo costituito dalla faccia in rilievo del parallelepipedo che costituiva il carattere mobile metallico. Nella figura 11.1 è riportato uno di questi blocchetti e vi si può riconoscere il disegno in rilievo e rovesciato di una 'm'.

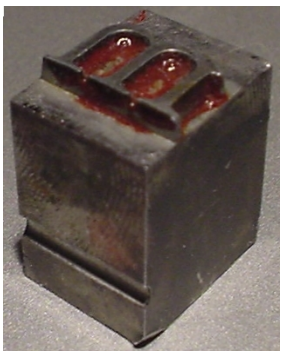

Figura 11.1: Il carattere metallico della ‘m’; il disegno, ovviamente, è come allo specchio e la sinistra e la destra sono scambiate

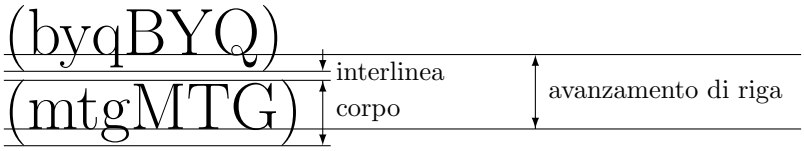

Figura 11.2: Relazioni fra il corpo, l’interlinea e l’avanzamento di riga nel caso di caratteri elettronici

interlinea striscia di metallo di un dato spessore da inserire fra due righe di testo composte con caratteri metallici; per estensione spazio verticale da aggiungere all’altezza di una riga composta con caratteri elettronici.

avanzamento di riga distanza fra le righe di base di due righe di testo composte con caratteri dello stesso corpo. Si può pensare a questa grandezza come alla somma del corpo e dell’interlinea; numericamente si tratta dello stesso valore, ma questa somma parte dalla base del rettangolo della faccia porta carattere, mentre l’avanzamento di riga parte dalla linea di base e raggiunge la linea di base della riga precedente; si veda la figura 11.2

x-height è l’altezza delle lettere minuscole senza ascendenti né discendenti, cioè ‘a c e m n o r s u v w x z’; questa altezza definisce anche l’unità di misura ‘ex’ che permette di prendere le misure in termini di altezza dei font che sono in uso. Il termine italiano per x-height è *occhio*.

quadro o quadrato era un blocco quadrato usato come spaziatore nel comporre con i caratteri metallici; le due facce del quadrato erano lunghe quanto il font in uso, se non altro perché l’altezza del quadrato doveva essere uguale al corpo. Con i font elettronici si conviene che la dimensione di ciascun lato del quadrato sia pari alla larghezza della ‘M’ per cui si usa parlare della distanza di una M, em in inglese. Se si leggono i dati forniti in calce alle tabelle 11.1–11.7, si vede che 1 em spesso non coincide affatto con il corpo anzi ne è sensibilmente diverso. Nonostante ciò questa spaziatura di un quadrato continua a fare riferimento all’unità em. In italiano il blocchetto si chiamava quadrato, e un blocchetto rettangolare largo il doppio di un quadrato si chiamava ‘quadratone’; i comandi per realizzare queste

spaziature si sono già incontrati negli esempi matematici; una spaziatura di un quadrato si ottiene con il comando `\quad` e di un quadratone con il comando `\qquad`.

legatura è l'operazione per la quale lettere distinte devono essere accostate in modo così forte per ottenere il giusto effetto estetico che vengono a formare un unico segno; si pensi a 'fi', 'ff', 'ffi', 'fl', 'ffl' presenti in ogni font latino; ma queste non sono le sole legature; scrivere `--` produce – e scrivere `---` produce —; si tratta sempre di legature; anche la lettera tedesca ß è il risultato della legatura di una 's lunga' e di una 's corta', in definitiva di due lettere 's' di forma diversa. In Italia non si usa più questo segno da alcuni secoli, ma i prototipografi italiani nel quattrocento e nel cinquecento la usavano abbondantemente; l'uso è cessato con la sparizione della 's lunga' nel diciottesimo secolo; questo segno assomigliava ad una lettera 'f', solo che la barretta non attraversava il fusto; se si esamina il segno (corsivo) della *ß*, si riconosce che la parte di sinistra è proprio una 's lunga'. Questa forma della lettera 's' veniva usata sempre come lettera iniziale o come lettera mediana, mentre la 's corta', quella che usiamo oggi, era in posizione terminale o come secondo elemento della doppia 's', da cui la legatura.

grazie sono quei piccoli tratti alla fine delle aste spesse o sottili che permettono, in particolare, di allineare esattamente i singoli caratteri sulla linea di base, riempiono leggermente lo spazio fra un carattere e l'altro e danno l'idea di una linea più compatta, così che l'occhio la segue meglio e ritrova più facilmente l'inizio della riga seguente quando l'occhio deve andare a capo. I caratteri con grazie sono più adatti alla lettura continua, mentre i caratteri senza grazie affaticano il lettore se egli dovesse leggere con continuità molte pagine di seguito. Caso a sé rappresenta il font Optima, disegnato da Hermann Zapf. Tale carattere, pur non avendo grazie, non può essere definito *sans serif* in virtù della sua particolare conformazione, che lo fa sembrare dotato di grazie.

sans serif è l'espressione inglese per designare i font senza grazie; in italiano si chiamano preferibilmente caratteri 'lineari', o 'bastone', o 'bastoncino'.

maiuscoletto è una raccolta di segni dove l'alfabeto è composto dalle lettere maiuscole e da una serie di 'minuscole' costituite da segni con lo stesso disegno delle maiuscole ma di altezza e larghezza minori; non si tratta di maiuscole rimpicciolite, ma di lettere disegnate con le stesse caratteristiche grafiche; per esempio hanno lo stesso spessore delle aste corrispondenti; una 'A' rimpicciolita alle dimensioni di una 'a' appare così: 'a'; affiancate appaiono così: aa; si vede benissimo che la 'a' rimpicciolita è comparativamente più chiara della a maiuscoletta.

proporzionale è il tipo di carattere che destina uno spazio diverso a ciascuna lettera; come ognuno può constatare la larghezza della 'm' minuscola è decisamente più grande della 'i' minuscola; non confrontiamo nemmeno la differenza fra maiuscole e minuscole perché è troppo evidente. La maggior parte, per non dire la totalità, dei caratteri usati in tipografia sono caratteri proporzionali. Tuttavia esiste ed è utilissimo, specialmente nei

testi dove si parla di informatica, poter disporre di un font a spaziatura fissa come quello usato in questo testo per descrivere i comandi usati per il mark-up di LaTeX; si osservino i due alfabeti maiuscolo e minuscolo sovrapposti:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
```

per rendersi conto di che cosa sia un carattere non proporzionale confrontandolo con i due corrispondenti alfabeti di un font proporzionale:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

famiglia rappresenta una collezione di font di diversi disegni ma con una caratteristica comune; le famiglie più importanti per LaTeX sono quella dei caratteri a spaziatura fissa, chiamati *typewriter type*, che costituiscono la famiglia identificata dall'istruzione `\ttfamily` o dal comando `\texttt`; la famiglia dei font proporzionali senza grazie, *sans serif*, identificata dall'istruzione `\sffamily` o dal comando `\textsf`; la famiglia dei font proporzionali con grazie, *roman*, identificata dall'istruzione `\rmfamily` o dal comando `\textrm`.

serie è la caratteristica di ogni famiglia di font di apparire più o meno scura; la tipografia tradizionale prevede diversi livelli di nero che vanno dal chiarissimo al nerissimo; per LaTeX si distinguono tre livelli di nero; il medio, il nero e il nero esteso; di norma il nero 'semplice' è assai raro nelle famiglie di caratteri usate di default da LaTeX, perciò le istruzioni per scegliere queste caratteristiche di default sono solamente `\mdseries` e `\bfseries` oppure i comandi `\textmd` e `\textbf`; non tutte le famiglie dispongono di serie più scure di quella che è considerata media.

forma rappresenta il tipo di disegno di una serie di caratteri della stessa famiglia e della stessa serie; per LaTeX le forme disponibili normalmente sono la forma diritta, *upright*, la forma corsiva, *italics*, e quella maiuscoletta, *small caps*; le istruzioni per scegliere queste forme sono `\upshape`, `\itshape` e `\scshape`, mentre i comandi sono `\textup`, `\textit` e `\textsc`. Sono disponibili anche la forma tondo inclinato, *slanted*, e il corsivo diritto, *upright italics*; queste in Italia sono meno usate, tuttavia LaTeX mette a disposizione l'istruzione `\slshape` e il comando `\textsl` per il tondo inclinato. Invece per il corsivo diritto non sono disponibili né l'istruzione né il comando, sebbene non sia difficile servirsi di questa forma con comandi di livello più basso.

Si confronti il tondo inclinato con il corsivo: *abcd* e *abcd*; si nota che si tratta di due disegni (diversamente) inclinati, ma sono completamente diversi; questo è il motivo per il quale il tondo inclinato è poco usato. Il corsivo diritto appare così: abcd.

codifica è la speciale modalità informatica con la quale un particolare segno di un dato font viene identificato nel file che contiene tutti i segni di quel particolare font. LaTeX conosce di default diverse codifiche, le più usuali

per chi scrive in italiano sono la codifica `OT1` e la codifica `T1`; la prima identifica i file che contengono 128 caratteri 'latini', la seconda identifica i file che contengono 256 caratteri 'latini'; i font 'latini' contengono oltre alle cifre, ai segni di interpunzione e ad un certo numero di segni speciali, gli alfabeti minuscolo e maiuscolo delle 26 lettere 'latine':

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

I numerosi segni accentati che compaiono nelle varie lingue che usano i caratteri latini sono ottenuti sovrapponendo l'accento alla lettera con la codifica `OT1`, mentre sono ottenuti con i disegni delle lettere già accentate con la codifica `T1`.

Per capire meglio il concetto di codifica si esaminino le tabelle 11.1–11.7 che contengono i caratteri latini con codifica OT1, i caratteri latini con codifica T1, i caratteri non letterali del Text Companion Font con codifica TS1, i caratteri cirillici con codifica OT2, i caratteri greci con codifica LGR.

In ogni casella di quelle tabelle c'è il segno che corrisponde alla posizione indicata dal numero decimale che compare nella stessa casella; i numeri preceduti da un solo apice o dal doppio apice nelle righe e nelle colonne esterne rappresentano parti degli indirizzi in notazione ottale o in notazione esadecimale; per esempio nella tabella 11.4 la lettera 'Í' ha un indirizzo decimale 205; questa casella si trova all'incrocio della riga con indirizzo ottale `'300` e della colonna con indirizzo ottale `'15` da cui si ricava l'indirizzo ottale complessivo eseguendo la somma dei due valori ottali: `'300 + '15 = '315`. Nella stessa maniera si ricava l'indirizzo complessivo esadecimale. Nei file interni di LaTeX sono quasi sempre usati gli indirizzi ottali o quelli esadecimali; quindi è conveniente disporre di tabelle che contengano tutti e tre i tipi di indirizzi.

Se si confrontano le tabelle 11.1 e 11.4, che si riferiscono entrambe ai caratteri latini, oltre alle ovvie differenze relative alle lettere accentate, ci sono alcuni segni uguali che si trovano ad indirizzi diversi; per esempio il segno 'ffi' si trova all'indirizzo 14 con la codifica OT1, ma all'indirizzo 30 con la codifica T1.

È assai raro dover cambiare codifica per comporre un testo; per questo LaTeX non dispone di una istruzione o di un comando di uso corrente per eseguire questo cambio di codifica, ma questo tipo di azione viene svolto dietro le quinte dalle istruzioni contenute nei file di servizio, che provvedono direttamente a cambiare l'alfabeto e la codifica a seconda della lingua che si usa. L'unico momento in cui il compositore decide di usare una delle due codifiche per i caratteri latini è all'inizio del file, nel prologo, quando ordina di usare il *font encoding* T1, la codifica T1, mediante la specifica

```
\usepackage[T1]{fontenc}
```

Se si desiderasse cambiare codifica in una posizione qualunque del testo bisognerebbe usare comandi di basso livello come `\fontencoding` e `\selectfont`, scrivendo per esempio

`\fontencoding{T1}\selectfont`

possibilmente all'interno di un gruppo per limitarne gli effetti al solo testo contenuto nel gruppo.

11.3 I comandi per la scelta dei font

Per scegliere i font il compositore dispone di una serie di istruzioni e di comandi che sono già stati quasi tutti esposti nel paragrafo precedente. Vale la pena di riepilogare.

11.3.1 La scelta del corpo e dell'avanzamento di riga

LaTeX dispone di una serie di comandi che si riferiscono al corpo dei caratteri in modo simbolico, invece che numerico; essi sono raccolti nella tabella 11.3 e indicano tutti quanti (in inglese) la relazione che il corpo prescelto ha rispetto al corpo normale.

Il corpo che funziona da corpo normale è quello di default, cioè di 10 pt, oppure quello che si specifica fra le opzioni del comando di apertura `\documentclass`; per altro fra queste opzioni, in generale, ci sono solo *11pt* e *12pt*. Alcune classi non standard offrono anche font normali di corpo più piccolo o più grande.

I corpi più piccoli di quello normale si differenziano di un punto l'uno dall'altro; i corpi più grandi sono invece in proporzione geometrica approssimativamente di ragione 1,2. Per il corpo normale di 10 pt i corpi specificati con i comandi della tabella 11.3 si susseguono così: 5 pt, 7 pt, 8 pt, 9 pt, 10 pt, 12 pt, 14,4 pt, 17,28 pt, 20,74 pt, 24,88 pt. Fuori della sequenza in progressione geometrica c'è anche il corpo di 10,95 pt, usato come corpo normale con l'opzione di *11pt*; si noti che questo corpo approssima la media geometrica fra i valori adiacenti 10 pt e 12 pt: $10{,}95 \approx \sqrt{10 \times 12}$.

Gli avanzamenti di riga sono prefissati a circa il 20% in più del corpo a seconda delle classi dei documenti. Se proprio si desidera allargare o restringere l'avanzamento di riga rispetto al valore di default (a parte scriversi un file di classe personalizzato) si può usare il comando `\linespread` il cui argomento funziona da moltiplicatore dell'avanzamento di default. Per limitarne l'efficacia bisogna dare questo comando dentro ad un ambiente oppure dentro a un gruppo. Si tenga però presente che l'effetto estetico combinato del corpo e dell'avanzamento di riga, insieme all'effetto che questi valori hanno sulla comodità di lettura, sono estremamente delicati e solo un grafico editoriale sa come modificare i valori di default di frazioni di punto percentuale; se un dilettante si mette a giocare con `\linespread` sappia che non è consigliabile superare mai il 5% in più o in meno; valori maggiori producono di solito risultati pessimi.

11.3.2 La scelta delle altre caratteristiche

Le altre caratteristiche dei font, tranne la codifica, che, come si è detto non viene cambiata quasi mai o viene cambiata automaticamente quando si cambia alfabeto, possono essere cambiate autonomamente l'una dall'altra con i comandi già descritti; se una certa combinazione di famiglia, serie e forma non è disponibile, LaTeX provvede da solo a sostituire il font mancante con un font di default o

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7	Φ 8	Ψ 9	Ω 10	ff 11	fi 12	fl 13	ffi 14	ffl 15	"00
’020	ı 16	ȷ 17	` 18	´ 19	˘ 20	¯ 21	˚ 22	¸ 23	ß 24	æ 25	œ 26	ø 27	Æ 28	Œ 29	Ø 30	Å 31	"10
’040	␣ 32	! 33	” 34	# 35	$ 36	% 37	& 38	’ 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	¡ 60	= 61	¿ 62	? 63	"30
’100	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79	"40
’120	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	“ 92	] 93	^ 94	˙ 95	"50
’140	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111	"60
’160	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	– 123	— 124	˝ 125	~ 126	¨ 127	"70
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,00000	x-height	4,30554 pt
spazio interparola	3,33332 pt	larghezza del quadrato	10,19999 pt
allungamento interparola	1,66665 pt	spazio extra	1,11111 pt
accorciamento interparola	1,11111 pt	corpo nominale	10,00000 pt

Tabella 11.1: Il font latino a 128 caratteri con codifica OT1

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	Њ 0	Љ 1	Ц 2	Э 3	І 4	Є 5	Ъ 6	Ћ 7	њ 8	љ 9	ц 10	э 11	і 12	є 13	ђ 14	ћ 15	"00
’020	Ю 16	Ж 17	Й 18	Ё 19	Ѵ 20	Ѳ 21	S 22	Я 23	ю 24	ж 25	й 26	ё 27	ѵ 28	ѳ 29	s 30	я 31	"10
’040	¨ 32	! 33	” 34	Ђ 35	˘ 36	% 37	´ 38	’ 39	(40	) 41	* 42	ђ 43	, 44	- 45	. 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	« 60	ı 61	» 62	? 63	"30
’100	˘ 64	А 65	Б 66	Ц 67	Д 68	Е 69	Ф 70	Г 71	Х 72	И 73	Ј 74	К 75	Л 76	М 77	Н 78	О 79	"40
’120	П 80	Ч 81	Р 82	С 83	Т 84	У 85	В 86	Ш 87	Щ 88	Ы 89	З 90	[91	“ 92	] 93	Ь 94	Ъ 95	"50
’140	‘ 96	а 97	б 98	ц 99	д 100	е 101	ф 102	г 103	х 104	и 105	ј 106	к 107	л 108	м 109	н 110	о 111	"60
’160	п 112	ч 113	р 114	с 115	т 116	у 117	в 118	щ 119	ш 120	ы 121	з 122	– 123	— 124	№ 125	ь 126	ъ 127	"70
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,00000	x-height	4,30554 pt
spazio interparola	3,77774 pt	larghezza del quadrato	11,55450 pt
allungamento interparola	1,74996 pt	spazio extra	1,16665 pt
accorciamento interparola	1,16665 pt	corpo nominale	10,00000 pt

Tabella 11.2: Il font cirillico a 128 caratteri con codifica OT2

Istruzione	Esempio
`\tiny`	ABCDEFG abcdefg
`\scriptsize`	ABCDEFG abcdefg
`\footnotesize`	ABCDEFG abcdefg
`\small`	ABCDEFG abcdefg
`\normalsize`	ABCDEFG abcdefg
`\large`	ABCDEFG abcdefg
`\Large`	ABCDEFG abcdefg
`\LARGE`	ABCDEFG abcdefg
`\huge`	ABCDEFG abcdefg
`\Huge`	ABCDEFG abcdefg

Tabella 11.3: Istruzioni per la scelta del corpo dei caratteri

con un font che abbia il maggior numero possibile delle caratteristiche richieste; nella tabella 11.5 vengono illustrate quasi tutte le combinazioni. Si noterà che in alcune caselle il contenuto è rosso; LaTeX ha sostituito il font mancante con un font che abbia un certo numero di caratteristiche fra quelle richieste.

Nella tabella 11.5 compaiono solo le istruzioni per assegnare una certa caratteristica al font scelto. Fra le altre cose si è indicata l'istruzione `\uishape` che in LaTeX non esiste; la si è definita mediante un comando, che verrà descritto meglio in seguito, in modo da ricorrere a istruzioni di livello inferiore:

```
\newcommand{\uishape}{\fontshape{ui}\selectfont}
```

Le istruzioni usate nella composizione della tabella possono essere usate anche durante la composizione del testo; tuttavia per comporre singole parole o brevi frasi è preferibile ricorrere ai comandi; tutti sono della forma `\text`⟨*xx*⟩, dove ⟨*xx*⟩ sono le stesse due lettere che compaiono all'inizio delle istruzioni; avremo quindi `\textrm`, `\textsl`, `\textbf`, `\textsf`, eccetera.

La sintassi di tutti questi comandi è la seguente:

`\text`⟨*xx*⟩`{`⟨*testo*⟩`}`

Perché è preferibile usare i comandi rispetto alle istruzioni? Perché con il comando il programma LaTeX può conoscere dove comincia e dove finisce il testo da comporre con una particolare caratteristica e quindi sa dove inserire la 'correzione italica', una piccola correzione per allontanare o per avvicinare i segni che precedono o seguono il ⟨*testo*⟩. Spesso i caratteri che seguono sono dei segni di interpunzione 'alti' come i due punti o il punto e virgola, oppure come il punto esclamativo o interrogativo; talvolta si tratta di parentesi; si noti allora l'effetto di chiudere fra parentesi una parola scritta in corsivo se non è

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	` 0	´ 1	ˆ 2	˜ 3	¨ 4	˝ 5	° 6	ˇ 7	˘ 8	¯ 9	˙ 10	¸ 11	˛ 12	’ 13	‹ 14	› 15	"00
’020	“ 16	” 17	„ 18	« 19	» 20	– 21	— 22	23	ı 24	ȷ 25	ff 26	fi 27	fl 28	ffi 29	ffl 30	31	"10
’040	␣ 32	! 33	" 34	# 35	$ 36	% 37	& 38	’ 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63	"30
’100	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79	"40
’120	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95	"50
’140	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111	"60
’160	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	- 127	"70
’200	Ă 128	Ą 129	Ć 130	Č 131	Ď 132	Ě 133	Ę 134	Ğ 135	Ĺ 136	Ľ 137	Ł 138	Ń 139	Ň 140	Ŋ 141	Ő 142	Ŕ 143	"80
’220	Ř 144	Ś 145	Š 146	Ş 147	Ť 148	Ţ 149	Ű 150	Ů 151	Ÿ 152	Ź 153	Ž 154	Ż 155	IJ 156	İ 157	đ 158	§ 159	"90
’240	ă 160	ą 161	ć 162	č 163	ď 164	ě 165	ę 166	ğ 167	ĺ 168	ľ 169	ł 170	ń 171	ň 172	ŋ 173	ő 174	ŕ 175	"A0
’260	ř 176	ś 177	š 178	ş 179	ť 180	ţ 181	ű 182	ů 183	ÿ 184	ź 185	ž 186	ż 187	ij 188	¡ 189	¿ 190	£ 191	"B0
’300	À 192	Á 193	Â 194	Ã 195	Ä 196	Å 197	Æ 198	Ç 199	È 200	É 201	Ê 202	Ë 203	Ì 204	Í 205	Î 206	Ï 207	"C0
’320	Ð 208	Ñ 209	Ò 210	Ó 211	Ô 212	Õ 213	Ö 214	Œ 215	Ø 216	Ù 217	Ú 218	Û 219	Ü 220	Ý 221	Þ 222	SS 223	"D0
’340	à 224	á 225	â 226	ã 227	ä 228	å 229	æ 230	ç 231	è 232	é 233	ê 234	ë 235	ì 236	í 237	î 238	ï 239	"E0
’360	ð 240	ñ 241	ò 242	ó 243	ô 244	õ 245	ö 246	œ 247	ø 248	ù 249	ú 250	û 251	ü 252	ý 253	þ 254	ß 255	"F0
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,00000	x-height	4,30450 pt
spazio interparola	3,33251 pt	larghezza del quadrato	9,99755 pt
allungamento interparola	1,66625 pt	spazio extra	1,11083 pt
accorciamento interparola	1,11083 pt	corpo nominale	10,00000 pt

Tabella 11.4: Il font latino a 256 caratteri con codifica T1

		\upshape	\slshape	\itshape	\scshape	\uishape
	\rmfamily	abcABC	*abcABC*	*abcABC*	ABCABC	abcABC
\mdseries	\sffamily	abcABC	*abcABC*	*abcABC*	ABCABC	abcABC
	\ttfamily	abcABC	*abcABC*	*abcABC*	ABCABC	*abcABC*
	\rmfamily	**abcABC**	***abcABC***	***abcABC***	**abcABC**	abcABC
\bfseries	\sffamily	**abcABC**	**abcABC**	**abcABC**	**abcABC**	abcABC
	\ttfamily	abcABC	abcABC	*abcABC*	abcABC	*abcABC*

Tabella 11.5: Le varie combinazioni di serie e di forma per le varie famiglie standard dei font usabili con LaTeX. I caratteri rossi sono stati sostituiti con altri caratteri aventi il maggior numero possibile delle caratteristiche richieste. Con i font latini aventi la codifica T1 si hanno meno sostituzioni; solo la famiglia *typewriter type* continua a non disporre della serie nera.

presente o se è presente la correzione italica: (XYZ) a confronto con (XYZ). Chiaramente il secondo caso è stato composto con il comando e il primo con l'istruzione; nel primo la parentesi chiusa batte contro la Z nel secondo no[1]. Si osservi infine che il caso più frequente di necessità della correzione italica si ha quando si usa il carattere corsivo; tuttavia il fenomeno si produce anche con il tondo[2]: fF e fF.

Fra i comandi per scegliere la forma dei segni di un font non bisogna scordare l'istruzione `\em` e il comando `\emph`; essi servono per evidenziare (in inglese *emphasize*) una parola o una breve locuzione all'interno di altro testo. Questa istruzione e questo comando hanno la proprietà di cambiare inclinazione al font (eventualmente passando dal tondo al corsivo, piuttosto che al tondo inclinato), per cui in questo testo scritto in tondo la parola *emphasize* è evidenziata in corsivo, mentre *in questo testo composto in corsivo* emphasize *viene evidenziato in tondo*. In entrambi i casi si è scritto `\emph{emphasize}`.

11.4 Altri font diversi da quelli di default

Talvolta è necessario usare font diversi da quelli di default vuoi per ottenere effetti diversi, per uniformarsi allo stile di documenti scritti con altri metodi, per aumentare la leggibilità, per rendere più compatto il testo, eccetera.

Sono stati predisposti diversi pacchetti per sostituire le famiglie di font usate di default con altre famiglie, generalmente con lo scopo di usare famiglie di font commerciali, anche se disponibili gratuitamente. Sul mercato, infatti, sono disponibili migliaia di famiglie di font. Essi costituiscono la croce e delizia dei disegnatori editoriali e richiedono, appunto, la loro professionalità per usarli al meglio. Per i 'dilettanti' può essere difficile usare i font commerciali. La presenza dei pacchetti per l'uso di questi font rende l'operazione molto più facile e l'uso di questi font diventa alla portata di tutti.

Si possono dunque invocare diversi pacchetti che servono per sostituire una alla volta le tre famiglie principali di font; tuttavia la cosa non è semplice, o meglio è semplicissimo con LaTeX, ma è difficile eseguire scelte in cui le famiglie si accordino alla perfezione, in particolare per quel che riguarda l'altezza delle minuscole, la famosa *x-height*.

Il testo accanto è composto con i font Computer Modern

Per questo scopo, allora si consiglia ai lettori di fare riferimento a due pacchetti in particolare: **txfonts** e **pxfonts**.

Il testo accanto è composto con i font Times extended

Il pacchetto **txfonts** consente di usare i font Times per il testo con grazie, il font Helvetica per il testo scritto senza grazie, e una variante del Courier per la famiglia a spaziatura fissa. I font sono caricati con fattori di scala predefiniti in modo che le altezze *x-height* siano il più possibile simili, non esattamente identiche, perché la forma dei segni richiede delle curve leggermente abbondanti specialmente con l'Helvetica.

La 'x' presente nel nome di questo pacchetto richiama il fatto che si tratta di una collezione estesa; infatti con l'uso di questi font diventano disponibili non solo tutti i simboli di LaTeX standard arricchito dai font del pacchetto **amsmath** ma una ulteriore varietà che ne aumenta ancora la versatilità. Quindi **tx** sta per 'Times eXtended'.

[1] Si noti che in tipografia si usano le parentesi non inclinate anche quando esse racchiudono testo scritto con caratteri inclinati.

[2] L'esempio non è cervellotico: fF è il simbolo dell'unità di misura 'femtofarad' che si incontra spesso in microelettronica.

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	` 0	´ 1	ˆ 2	˜ 3	¨ 4	˝ 5	° 6	ˇ 7	˘ 8	¯ 9	· 10	¸ 11	˛ 12	’ 13	14	15	"00
’020	16	17	" 18	19	20	— 21	— 22	23	← 24	→ 25	⁀ 26	⁀ 27	⁀ 28	⁀ 29	30	31	"10
’040	ƀ 32	33	34	35	$ 36	37	38	’ 39	40	41	* 42	43	, 44	= 45	. 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	58	59	⟨ 60	— 61	⟩ 62	63	"30
’100	64	65	66	67	68	69	70	71	72	73	74	75	76	Ʊ 77	78	◯ 79	"40
’120	80	81	82	83	84	85	86	Ω 87	88	89	90	⟦ 91	92	⟧ 93	↑ 94	↓ 95	"50
’140	` 96	97	⋆ 98	o|o 99	† 100	101	102	103	104	105	106	107	108	ꝏ 109	♪ 110	111	"60
’160	112	113	114	115	116	117	118	119	120	121	122	123	124	125	~ 126	= 127	"70
’200	◡ 128	◡ 129	″ 130	‶ 131	† 132	‡ 133	‖ 134	‰ 135	• 136	°C 137	$ 138	¢ 139	f 140	₡ 141	₩ 142	₦ 143	"80
’220	₲ 144	₱ 145	£ 146	₨ 147	⁉ 148	¡ 149	₫ 150	™ 151	‰ 152	¶ 153	฿ 154	№ 155	℅ 156	℮ 157	○ 158	℠ 159	"90
’240	{ 160	} 161	¢ 162	£ 163	¤ 164	¥ 165	¦ 166	§ 167	¨ 168	© 169	a 170	ↄ 171	¬ 172	℗ 173	® 174	¯ 175	"A0
’260	° 176	± 177	2 178	3 179	´ 180	µ 181	¶ 182	· 183	※ 184	1 185	o 186	√ 187	1/4 188	1/2 189	3/4 190	€ 191	"B0
’300	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	"C0
’320	208	209	210	211	212	213	× 214	215	216	217	218	219	220	221	222	223	"D0
’340	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	"E0
’360	240	241	242	243	244	245	÷ 246	247	248	249	250	251	252	253	254	255	"F0
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,00000	x-height	4,30450 pt
spazio interparola	3,33251 pt	larghezza del quadrato	9,99755 pt
allungamento interparola	1,66625 pt	spazio extra	1,11083 pt
accorciamento interparola	1,11083 pt	corpo nominale	10,00000 pt

Tabella 11.6: Il Text Companion Font con codifica TS1

Il testo accanto è composto con i font Palatino extended

L'altro pacchetto **pxfonts** mette a disposizione il Palatino esteso; il Palatino viene usato al posto del Times, ma per le altre famiglie si usano ancora l'Helvetica e il Courier. In questo caso i fattori di scala con cui sono invocati i font accessori sono diversi da quelli del Times, perché il Palatino a pari corpo appare decisamente più grande e più largo

Il testo accanto è nuovamente composto con i font Computer Modern

Il lettore confronti i capoversi marcati con le note marginali; si tratta di tre testi composti con font dello stesso corpo (10 pt) e con lo stesso avanzamento di riga (12 pt), ma con collezioni diverse di font. Eventualmente usi una lente di ingrandimento (anche quella fornita dal programma di visualizzazione del file PDF) per esaminare i dettagli di forma delle lettere; inoltre noti il differente livello di nero dei testi composti con questi font.

11.5 Il Text Companion Font

LaTeX consente di usare un numero enorme di simboli, specialmente in matematica; ma da solo LaTeX non offre i simboli che si usano nel testo, dai segni delle unità monetarie ad altri segni che possono trovare posto in scritti di vario genere.

Così quando furono prodotti i primi font latini con la codifica T1, fu sentita immediatamente la necessità di un certo numero di segni o comandi non coperti con i simboli di nessun altro font, e fu predisposto il Text Companion Font, i cui simboli sono riportati nella tabella 11.6; ritorna per l'ennesima volta il segno del mho ℧, il cui unico pregio è quello di poter essere scritto in modo testo con il comando `\textmho`. Ahimè, trattandosi di un segno bandito dalle norme ISO, questo segno appare per la terza volta, ma inutilmente, perché non può essere usato.[3]

Il Text Companion Font viene caricato e reso utilizzabile caricando il pacchetto **textcomp**.

11.6 Gli alfabeti diversi da quello latino

Per scrivere in altre lingue bisogna avere invocato il pacchetto `babel` avendo indicato fra le opzioni le lingue che si vogliono usare; l'ultima lingua indicata nell'elenco delle opzioni diventa la lingua di default per l'intero documento.

Quando si invoca una qualsiasi lingua che si scriva con l'alfabeto cirillico, russo, bulgaro, ucraino, eccetera, il pacchetto `babel` imposta direttamente anche l'uso dell'alfabeto cirillico. Una versione a 128 caratteri è riportata nella tabella 11.2, però con alcune di quelle lingue può venire scelto un altro encoding, magari a 256 caratteri.

Quando si invoca la lingua greca senza ulteriori specificazioni, viene anche impostato l'alfabeto greco riportato nella tabella 11.7 ma predisposto per la scrittura monotonica moderna, cioè con gli accenti ridotti al solo accento acuto nelle parole polisillabiche e dove sono eliminati gli spiriti; con queste semplificazioni diventa più frequente l'uso della dieresi per indicare gruppi di vocali che non formano dittonghi. Se si specifica invece l'attributo `polutoniko` per la lingua greca ecco che diventano disponibili tutti i segni indicati nella tabella 11.7.

[3] In realtà è ancora usato negli Stati Uniti dove le norme ISO hanno meno seguito che in Europa. In Europa, invece, non dovrebbe più esserci nessuno che lo usi.

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	– 0	◌͡ 1	𝔸 2	ℍ 3	𝕏 4	𝕄 5	ҁ 6	ҁ 7	ı 8	A_{I} 9	H_{I} 10	Ω_{I} 11	A 12	Ϋ 13	α 14	ϋ 15	"00
’020	◌́ 16	◌̀ 17	˩ 18	ˤ 19	◌̆ 20	Ϙ 21	Ϟ 22	Ϡ 23	€ 24	‰ 25	ə 26	ɔ 27	‘ 28	’ 29	◌̆ 30	– 31	"10
’040	◌̋ 32	! 33	’ 34	◌̊ 35	◌̊ 36	% 37	· 38	′ 39	(40	) 41	* 42	+ 43	, 44	- 45	· 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	· 59	◌̡ 60	= 61	◌̓ 62	; 63	"30
’100	◌̔ 64	A 65	B 66	◌̓ 67	Δ 68	E 69	Φ 70	Γ 71	H 72	I 73	Θ 74	K 75	Λ 76	M 77	N 78	O 79	"40
’120	Π 80	X 81	P 82	Σ 83	T 84	Υ 85	◌̋ 86	Ω 87	Ξ 88	Ψ 89	Z 90	[91	◌̔ 92	] 93	◌̋ 94	◌̊ 95	"50
’140	◌̀ 96	α 97	β 98	ς 99	δ 100	ε 101	φ 102	γ 103	η 104	ι 105	ϑ 106	ϰ 107	λ 108	μ 109	ν 110	ο 111	"60
’160	π 112	χ 113	ρ 114	σ 115	τ 116	υ 117	118	ω 119	ξ 120	ψ 121	ζ 122	« 123	· 124	» 125	◌̃ 126	— 127	"70
’200	◌̀ 128	◌́ 129	◌̇ 130	◌̉ 131	◌̡ 132	◌̡́ 133	◌̡̇ 134	◌̡̉ 135	◌́ 136	◌̆ 137	◌̆ 138	◌̃ 139	◌́ 140	◌̆́ 141	◌̆́ 142	◌̆̇ 143	"80
’220	◌̃ 144	◌̆ 145	◌̆ 146	F 147	◌̃̆ 148	◌̆́ 149	◌̆́ 150	◌͡ 151	◌̀ 152	◌̇ 153	◌̇ 154	155	◌̀ 156	◌́ 157	◌̇ 158	159	"90
’240	◌́ 160	◌̋ 161	◌̋ 162	◌̏ 163	◌́ 164	◌̋ 165	◌̋ 166	◌̏ 167	◌̃ 168	◌̋ 169	◌̃ 170	◌̏ 171	◌̃ 172	◌̋ 173	◌̋ 174	◌̏ 175	"A0
’260	◌̀ 176	◌́ 177	◌̇ 178	◌̉ 179	◌̡ 180	◌̡́ 181	◌̡̇ 182	◌̡̉ 183	ώ 184	◌̆ 185	◌̆ 186	◌̃ 187	◌́ 188	◌̆́ 189	◌̆́ 190	◌̆̇ 191	"B0
’300	◌̃ 192	◌̃ 193	◌̃ 194	F 195	◌̃ 196	◌̃́ 197	◌̃́ 198	199	◌̀ 200	◌́ 201	◌̇ 202	◌̂ 203	◌̀ 204	◌́ 205	◌̇ 206	◌̉ 207	"C0
’320	◌́ 208	◌̋ 209	◌̋ 210	◌̏ 211	◌́ 212	◌̆ 213	◌̆ 214	◌̉ 215	◌̃ 216	◌̋ 217	◌̋ 218	Ϊ 219	◌̃ 220	◌̆ 221	◌̆ 222	Ϋ 223	"D0
’340	◌̀ 224	◌́ 225	◌̇ 226	◌̉ 227	◌̀ 228	◌́ 229	◌̇ 230	◌̉ 231	◌́ 232	◌̋ 233	◌̆ 234	◌̃ 235	◌́ 236	◌̋ 237	◌̆ 238	◌̉ 239	"E0
’360	◌̈ 240	◌̀ 241	◌́ 242	◌̋ 243	◌̈ 244	◌̉ 245	◌́ 246	◌̃ 247	◌̣ 248	◌̡ 249	◌̣ 250	◌̡ 251	◌̣ 252	253	◌̓ 254	◌̓ 255	"F0
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,00000	x-height	4,30450 pt
spazio interparola	3,33251 pt	larghezza del quadrato	9,99755 pt
allungamento interparola	1,66625 pt	spazio extra	1,11083 pt
accorciamento interparola	1,11083 pt	corpo nominale	10,00000 pt

Tabella 11.7: Il font greco a 256 caratteri con codifica LGR

Per usare il greco bisogna leggere attentamente la documentazione `babel` relativa a questa lingua per usare correttamente la transcodifica della tastiera USA (o anche italiana) nei caratteri greci; si noti che per la moltitudine di accenti e altri segni diacritici presenti con molte vocali, sono previste numerose legature; così la sequenza `>~a|` produce il segno ᾷ.

Per produrre testi greci finemente commentati con le notazioni filologiche del settore si può estendere la funzionalità di LaTeX con il pacchetto **teubner** che consente di introdurre una grande varietà di segni specifici, di usare una serie di font greci inclinati particolarmente leggibili, sviluppati preso la tipografia Teubner di Lipsia e in onore della quale è stato scelto il nome del pacchetto. Tuttavia va fatto osservare che pacchetti di vario genere possono collidere; in particolare il pacchetto **teubner** sembra collidere con i pacchetti **amsmath** e compagni; è vero che difficilmente in un lavoro di filologia classica greca c’è bisogno di usare matematica avanzata, tuttavia questo problema esiste e in questi casi l’utente che si scontra con problemi derivanti dal conflitto di pacchetti dovrebbe avere ‘il senso civico’ di avvisare i curatori dei pacchetti, come avviene regolarmente con tutto il software libero, il cui progresso avviene proprio grazie a questa interazione fra programmatori e utenti.

11.7 La gestione dei font

Gestire la moltitudine di font sul proprio elaboratore può talvolta risultare faticoso; bisogna tenere presente che i font classici di LaTeX sono per lo più distribuiti sotto forma di sorgenti scritti con il linguaggio di programmazione METAFONT, e che tutti i programmi delle distribuzioni del sistema TeX sono in grado di lanciare METAFONT, il programma, in modo da generare al volo i font in formato bitmapped adatti al tipo di schermo e di stampante che si sono configurati durante l'installazione del sistema.

Tuttavia mentre su carta i font bitmapped creati per la densità di stampa della stampante in uso danno luogo ad un risultato perfetto, quegli stessi font non sono il meglio che si possa usare per leggere il testo sullo schermo, specialmente se si desidera leggere il testo con ingrandimenti diversi.

Sarebbe meglio disporre dei font in formato outline, ma questi sono difficili da ottenere dai file sorgente METAFONT, quindi bisogna scaricarli dalla rete 'già fatti'. Inoltre c'è la miriade di font che si possono comperare o che si trovano come freeware, sia in formato Adobe Type 1 (PostScript), sia in formato TrueType e in formato OpenType. La loro installazione non è banale per i seguenti motivi.

LaTeX per lavorare non ha bisogno dei disegni dei font, ma delle loro caratteristiche geometriche; queste sono raccolte in file con l'estensione `.tfm` (TeX Font Metrics); se si usa LaTeX sarà poi il driver che trasforma il file DVI in forma leggibile quello che avrà bisogno dei disegni che, in formato bitmapped, hanno estensioni del tipo `.`⟨*dpi*⟩`pk`, in formato PostScript `.pfb` (PostScript Font in Binary format), in formato TrueType `.ttf` (TrueType Font). Se invece si usa pdfLaTeX esso ha bisogno simultaneamente sia dei file metrici sia dei file outline o, in mancanza, dei file bitmapped.

I font bitmapped di origine METAFONT vengono generati insieme con il file `.tfm`, quindi non c'è bisogno d'altro.

I font PostScript ottenuti dal mercato o dalla rete (non ottenuti trasformando i font bitmapped, che ne sono già dotati) sono disponibili con il file metrico che ha formato diverso dal `.tfm` ed ha estensione `.afm` (Adobe Font Metrics).

I font TrueType non dispongono di un file metrico separato, perché le informazioni metriche sono già contenute nel file `.ttf`.

In questa situazione bisogna leggere attentamente le guide che accompagnano il sistema TeX della propria distribuzione e seguirle scrupolosamente; i programmi per le debite conversioni di formato dei file metrici fanno già parte della distribuzione, ma le operazioni da fare devono esser svolte dall'utente/amministratore del PC con grande cura.

La raccomandazione di leggere la documentazione in questo caso è più che una raccomandazione: è un obbligo!

Però questa difficoltà è compensata dal fatto che il sistema TeX, contrariamente a quel che si mormora da parte di coloro che non leggono la documentazione, non è affatto vincolato ai suoi font METAFONT, ma può lavorare con qualunque font esista al mondo.

Infine va ricordato che l'ultimo nato della famiglia dei programmi di composizione del sistema TeX, XeTeX, consente di lavorare con i font con la codifica UNICODE e apparentemente sa trovare da solo le informazioni metriche di cui necessita sia nei file `.tfm` o `.afm` sia all'interno dei file `.ttf` e quindi l'uso dei font e degli alfabeti più disparati (compresi quelli ad ideogrammi) diventa de-

cisamente semplificata, specialmente se si dispone di uno shell editor che possa scrivere il testo sorgente facendo uso direttamente della codifica UNICODE.

Va fatta una precisazione: il sistema TEX viene distribuito dalla rete o mediante dischi di installazione, ma generalmente l'utente installa solo la versione base arricchita dai pacchetti di estensione di cui necessita via via che usa questo sistema di composizione.

L'utente non necessariamente installa tutto perché il sistema e l'intera collezione di estensioni è enorme e quindi piuttosto ingombrante sul disco.

Questo è particolarmente vero per i pacchetti di estensione che arricchiscono le potenzialità compositive del sistema mediante un arricchimento di font; i diversi formati, però, fanno sì che siano necessarie delle conversioni, perché non tutti i programmi usano lo stesso formato.

Inoltre, come si è avuto occasione di commentare più volte, le codifiche dei font possono essere diverse, spesso lo sono; ma TEX conosce solo le sue codifiche, quindi, talvolta è necessario anche provvedere al cambiamento di codifica. Per questo scopo si usano sia i 'font virtuali' sia i 'vettori di codifica'. Il discorso, come si vede, può diventare assai complicato, e non si procede oltre.

Tuttavia, per rendersi conto perché certe cose non sono fatte come uno si aspetterebbe, bisogna ricordare che i file DVI possono essere mostrati su schermo da programmi come YAP o xdvi; e questi quando devono rappresentare graficamente i singoli segni dei font ricorrono di default alle loro immagini bitmapped.

Altri programmi non facenti parte in senso stretto del sistema TEX, come per esempio dvips (che trasforma i file dal formato DVI al formato PostScript) oppure dvipdfm (che trasforma i file dal formato DVI al formato PDF) preferiscono usare i font di tipo outline e ricorrono a quelli bitmapped solo in caso di necessità. Hanno perciò bisogno di sapere quali font outline siano disponibili e dove si trovino, perché questa informazione non è contenuta nel file DVI. pdfLATEX, invece, ha bisogno di conoscere subito quali font siano disponibili, in che formato e dove si trovino questi font, perché nel file sorgente non c'è nessun accenno a questa informazione. E, attenzione, non dimentichiamo che quanto è disponibile su una macchina non è necessariamente disponibile su un'altra, perché il suo amministratore non ha caricato gli stessi pacchetti e/o non ha comperato gli stessi prodotti commerciali.

Ecco allora che sono necessari tre file per svolgere le funzioni appena descritte; essi si chiamano `psfonts.map`, `dvipdfm.map` e `pdftex.map`; sono collocati in tre cartelle diverse dell'albero di sistema o dell'albero personale delle cartelle di TEX; essi contengono le opportune informazioni rispettivamente per dvips, o per dvipdfm, o per pdfLATEX.

Ogni volta che si installa una nuova collezione di font outline o anche dei file sorgente per la creazione di ulteriori font METAFONT di tipo bitmapped, bisogna aggiornare il contenuto di `psfonts.map` e compagni. Questa è una operazione delicata e va eseguita esattamente come dicono le istruzioni che accompagnano la distribuzione del sistema TEX, perché ovviamente tutto ciò dipende dalla particolare macchina e dal suo sistema operativo, anche se dal lato TEX si cerca di fare in modo che le operazioni da eseguire siano il più possibile simili, possibilmente uguali.

Ma durante la lavorazione di questo testo le persone che hanno contribuito si sono rese conto che nonostante tutto la loro configurazione, benché fortemente personalizzata e piuttosto ricca di estensioni, tuttavia rendeva difficile il lavoro

di gruppo, e qualche volta lo rendeva impossibile. Per questo si è scelto di usare i font Computer Modern con codifica OT1, anche se non è quella ottimale per le lingue con gli accenti come l'italiano, ma è sicuramente presente su ogni installazione.

11.7.1 Esempio di installazione di una collezione di font

Qui si cercherà di chiarire quanto specificato nel paragrafo precedente con un esempio pratico; ci accingiamo a configurare il sistema per poter scrivere in greco. Abbiamo bisogno di **babel** per poter gestire le lingue, ma in particolare abbiamo bisogno che sia attrezzato per il greco, oppure abbiamo bisogno di completarlo con le cose che riguardano questa lingua. Abbiamo bisogno dei file sorgente dei font METAFONT, anche se preferiremmo usare i font outline. Abbiamo perciò bisogno dei file outline dei font greci, della loro mappa e visto che ci siamo, anche dei file metrici.

Verifichiamo dapprima se il pacchetto **babel** installato nel nostro sistema sia già attrezzato per il greco; potremmo semplicemente lanciare LaTeX con un semplice file sorgente del tipo di questo:

```
% Testo di prova per il greco
\documentclass{minimal}
\usepackage[greek]{babel}
\begin{document}
Qa'ire!
\end{document}
```

Se LaTeX esegue la compilazione, e i font METAFONT sono già disponibili, sullo schermo del previewer appare la parola Χαίρε. Ma se la lingua greca non dispone di tutto ciò di cui necessita, allora possono essere successe diverse cose.

1. Subito all'inizio LaTeX si ferma dicendo che non dispone del file `greek.ldf`; questo indica che **babel** non è attrezzato per niente ai fini del greco. Bisogna installare una versione più recente e ben configurata.

2. LaTeX si lamenta che la codifica LGR gli è sconosciuta; **babel** è mal configurato per il greco, nel senso che ne conosce le specificità ma ne ignora i font; bisogna caricare i font greci con i vari file `.fd` di descrizione delle loro famiglie.

3. LaTeX completa il suo lavoro senza messaggi d'errore, ma il previewer non è in grado di rappresentare i font greci e forse scrive sullo schermo 'Xaíre' o qualcosa del genere. Vuol dire che LaTeX ha trovato i file `.tfm` dei font greci, ma il previewer non ha trovato né i file dei font rasterizzati, né ha trovato i file sorgente con estensione `.mf` per crearli; bisogna evidentemente caricare questi file sorgente.

4. Invece di usare LaTeX si è usato pdfLaTeX; tutto è andato bene ma il previewer del file PDF mostra la parola Χαίρε con contorni mal definiti, oppure essi appaiono definiti se li si guarda senza ingrandimento, ma appaiono tutti scalettati se li si guarda attraverso la lente di ingrandimento del previewer. Questo significa che mancano i font outline oppure che il file `psfonts.map` e i suoi compagni non sono configurati correttamente.

Come si vede le possibilità che inizialmente ci sia qualcosa che non va sono tante; non è un caso frequente, ma può succedere. Allora bisogna cercare i file mancanti e/o configurare correttamente il pacchetti esistenti o i file di configurazione. Dopo ogni pacchetto installato non ci si dimentichi mai di aggiornare il database dei nomi dei file lanciando texhash sui sistemi di tipo UNIX, o lanciando l'apposito wizard della distribuzione MiKTeX sulle macchine Windows. Questa operazione è essenziale; se il database dei nomi dei file non è aggiornato e completo nessun programma del sistema TEX può funzionare correttamente.

1. Il pacchetto **babel** fa normalmente parte della distribuzione del sistema TEX, quindi se non è in grado di gestire il greco, vuol dire che è molto vecchio; o se ne installa una versione nuova, oppure è meglio controllare che tutto il sistema TEX che si sta usando non sia troppo vecchio; il greco dovrebbe essere gestibile correttamente dagli ultimi anni '90, diciamo 1998, quindi il lettore valuti se la sua distribuzione è più vecchia e, nel caso, reinstalli completamente il sistema TEX. Non sembra possibile che una versione recente non sia in grado di gestire il greco, tuttavia non si può escludere che scaricando il pacchetto qualche file si sia corrotto; allora è meglio ricaricare tutto il pacchetto cercando su CTAN l'ultima versione; l'indirizzo in cui cercare è `www.ctan.org/tex-archive/macros/latex/required/babel/`; da questo sito si può scaricare l'intero pacchetto compresso in formato `.zip` oppure si possono scaricare le singole componenti del pacchetto; se si sceglie di scaricare l'intero pacchetto basta decomprimerlo e poi seguire le istruzioni del file `README` per sapere in quali cartelle mettere i vari file; se si sceglie di scaricare solo le componenti per il greco, basta scaricare `greek.dtx`, `greek.fdd`, `greek.ins` e, se si vuole, `athnum.dtx`. Lanciando LATEX su `greek.ins` si estraggono tutti i file necessari, mentre lanciando LATEX sui file con estensione `.dtx` e `.fdd` se ne ottiene la documentazione. Si devono poi spostare i file così ottenuti nelle cartelle di **babel** per avere i file mancanti oppure per sostituire i file corrotti.

2. Nel sito precedentemente indicato si trova anche il file `GreekFonts.txt` che contiene istruzioni varie per scaricare i font greci, in particolare i file sorgente per il programma METAFONT, e, già pronti, i file metrici `.tfm`. Se si seguono le indicazioni contenute in quel file si può scaricare tutta la collezione dei font greci rasterizzati o 'rasterizzabili'; in realtà non vengono scaricati i file `.pk`, ma, scaricando i file sorgente, il programma METAFONT è in grado di creare i file rasterizzati 'on demand', cioè quando occorrono.

3. Se si vogliono usare i font outline, bisogna scaricare anche i file `.pfb` relativi ai font greci. Beccari e Syropoulos inizialmente avevano predisposto tutti i file `.pfb` corrispondenti a tutti i corpi dei font rasterizzati. Nel 2005 Syropoulos ha deciso che l'insieme dei font `.pfb` era troppo ingombrante ed ha pensato che sarebbero bastati i font di corpo 10 pt, tanto, essendo font outline, questi possono essere ingranditi o rimpiccioliti a piacere. Egli quindi ha chiesto di eliminare dal sito CTAN tutti i font di corpo diverso da quello di 10 pt, e ha dato istruzioni di usare nel preambolo il pacchetto **type1ec** e di specificare nella chiamata del pacchetto:

   ```
   \usepackage[10pt]{type1ec}
   ```

In questo modo i vari font greci da 10 pt vengono scalati secondo necessità. Trattandosi di un font da usare essenzialmente solo per le parti testuali di un documento, e non per le parti matematiche, questa decisione è forse un po' discutibile, ma certamente accettabile. Sicuramente consente di scaricare dalla rete molte decine di megabyte in meno! Il sito `www.ctan.org/tex-archive/fonts/greek/cb/` consente di scaricare tutto quanto concerne i font greci; in particolare la cartella `www.ctan.org/tex-archive/fonts/greek/cb/type1/` contiene tutti i file dei font `.pfb`, mentre la cartella `www.ctan.org/tex-archive/fonts/greek/cb/dvips/` contiene il file `cbgreek.map` necessario per configurare i file `psfonts.map` e compagni.

Nei sistemi TeX aggiornati al 2007, conformi quindi alla nuova struttura delle cartelle a cui la distribuzione TeXlive si conforma, così come gli altri sistemi più o meno derivati da TeXlive, nel proprio albero personale è conveniente creare una cartella `.../fonts/map/greek` e qui copiarvi il file `cbgreek.map`, così nelle operazioni successiva, aggiornato il database dei nomi dei file, il file viene trovato senza difficoltà.

Analogamente i file metrici e i file dei font verranno collocati nelle cartelle `.../fonts/tfm/local/greek` e `.../fonts/type1/local/greek`; se queste cartelle non esistessero ancora, basta crearle con i comandi in linea disponibili con ogni sistema operativo, oppure ricorrendo alle opportune interfacce grafiche per la gestione delle cartelle.

4. Scaricati i file suddetti nelle opportune cartelle e aggiornato il database dei nomi dei file, bisogna procedere a configurare le mappe dei font outline. Si consiglia di cercare nell'albero di sistema `.../web2c/` il file `updmap.cfg` e di copiarlo nel proprio albero personale nella stessa sotto-cartella (se questa mancasse, la si crea). Si apre detto file con un editor testuale (va benissimo lo shell editor con il quale si gestiscono i file sorgente da far elaborare a LaTeX) e vi si aggiunge in fondo la riga[4]

   ```
   Map cbgreek.map
   ```

   rispettando le maiuscole. Successivamente si suggerisce di cambiare cartella di lavoro, spostandosi proprio in questa cartella `web2c` del proprio albero locale; si esegua poi il comando

   ```
   updmap ./updmap.cfg
   ```

   Con certi sistemi operativi potrebbe non essere necessario specificare `./`; con certi altri occorre usare il backslash; con altri ancora bisogna specificare l'indirizzo completo del comando eseguibile, ma questo ogni utente sa come gestirselo se conosce a dovere le funzionalità del proprio sistema operativo.

[4] All'inizio di questo file c'è un avviso che raccomanda di non editare direttamente questo file, ma di farlo con strumenti di configurazione che purtroppo sono diversi da sistema a sistema; onestamente lo scrivente ha editato direttamente questo file, una volta ricopiato nel proprio albero personale, senza incorrere in nessun problema; naturalmente seguire le indicazioni specificate all'inizio del file va sempre bene.

Durante l'esecuzione del programma `updmap` sullo schermo scorrono diversi messaggi che conviene poi eventualmente andare a rileggere, se l'operazione non fosse andata a buon fine. Supponendo, invece, che tutto si sia svolto correttamente, sul proprio albero di cartelle compaiono ora tre nuovi file[5]

```
.../fonts/map/dvipdfm/updmap/dvipdfm.map
.../fonts/map/dvips/updmap/psfonts.map
.../fonts/map/pdftex/updmap/pdftex.map
```

Aprendo uno qualunque di questi file con un editor testuale, vi si devono ritrovare, tra gli altri ma in ordine alfabetico, tutti i nomi dei font greci in formato `.pfb` seguiti dalle altre informazioni che il programma ha trovato nel file `cbgreek.map`.

5. Eseguite tutte queste operazioni ogni programma eseguibile che produca la compilazione o la trasformazione del file sorgente `.tex` *deve* poter trovare i font greci e deve poter lavorare con i file di descrizione dei font greci e **babel** non può lamentarsi di non conoscere la codifica LGR. Insomma tutto deve filare liscio; se ciò non succedesse bisogna ripetere scrupolosamente i passi indicati eliminando gli errori che sono stati commessi.

11.7.1.1 Altri font outline

In questo testo sono stati usati tra gli altri i font della famiglia Text Companion Font; i suoi file `.pfb` non sono caricati di default, ma l'installazione del sistema carica solamente i file sorgente per METAFONT.

Nel solito archivio CTAN nella cartella `www.ctan.org/tex-archive/fonts/ps-type1/cm-super/` si trovano i font della collezione CM-super che contengono sia la collezione Text Companion Font in formato outline, sia i font testuali della collezione EC, sia anche i font cirillici. Scaricare l'intera collezione sarebbe pesantissimo, come lo sarebbe stato per i font greci, comunque è una via percorribile, anche perché alcune distribuzioni Linux consentono di scaricare e installare il tutto con `apt-get`, senza bisogno di preoccuparsi d'altro.

Nella cartella `www.ctan.org/tex-archive/fonts/lm/` si trovano i font Latin Modern (LM) che pur non essendo definitivamente fissati, offrono una migliore collezione di segni outline, apparentemente migliori dei CM-super secondo gli intenditori. Certamente, se non si ha bisogno di usare il cirillico, questa collezione è molto più parca per quel che riguarda lo spazio sul disco.

I font della collezione Latin Modern comprendono anche i file di ricodifica conformi alle norme Adobe per i font PostScript e permettono di servirsi di codifiche diverse dalle solite. Tuttavia, nel momento di caricare il file di mappa, si consiglia di inserire nella propria versione personale del file `updmap.cfg` il nome del file `lm.map` (eventualmente decommentandolo, visto che di solito vi è già inserito). Si consiglia di non inserire il nome specifico corrispondente ad una particolare codifica, perché questi ultimi file specificano solo le transcodifiche dei font testuali, mentre non danno nessuna informazione sui font matematici;

[5] Con la distribuzione MacTEX basata su TEXlive i file compaiono dentro una specie di albero invisibile all'indirizzo `~/.texlive2007/texmf-var/`. Il file non è facile da trovare con il programma Finder del sistema operativo Mac OS X, ma se si usa il Terminal e si usano i comandi in linea, non è difficile trovare questi file nascosti; bisogna solo saperlo...

al contrario il file di mappa `lm.map` contiene tutte le informazioni relative alla collezione Latin Modern, con riferimento a tutte le codifiche esistenti, e senza dimenticare i font matematici.

Nella cartella `www.ctan.org/tex-archive/systems/vtex/common/` si trovano diversi file compressi in formato `.zip` che contengono diverse collezioni di font; tra l'altro sono presenti anche i font greci 'completi', cioè non solo quelli di corpo 10 pt.

Ma sempre nella stessa cartella si trova il file compresso `vtex-fonts2.zip` che contiene davvero i font EC e TS, la cui installazione dovrebbe risultare molto semplice, anche se nel file compresso la documentazione (per altro interessante) fa riferimento ancora ai font DC (i font EC in versione sperimentale); i file `.pfb` e i rispettivi file `.tfm` sono nelle apposite sottocartelle del file compresso. La generazione delle mappe non presenta difficoltà perché nel file `updmap.cfg` le righe relative ai font EC e ai font TS esistono già, salvo che eventualmente sono commentate, quindi non appaiono attive.

11.7.1.2 Installare un set di font da zero

Supponiamo di voler usare i font per i lettori ottici di caratteri, i font OCR. Se ne possono scaricare dagli archivi CTAN i file sorgente per METAFONT; questi sono i font da `ocrb5.mf` a `ocrb10.mf` più i file di servizio; non è difficile nel proprio albero di cartelle creare una nuova cartella `.../fonts/source/ocr/` e depositarvi i sorgenti `.mf`; naturalmente, avendo modificato le cartelle e/o i loro contenuti, bisogna aggiornare il database dei nomi dei file.

Se ci si accontenta dei file rasterizzati, questo è tutto quel che serve. Se si vogliono usare le versioni outline, bisogna crearsele.

Bisogna installare nel proprio calcolatore nelle apposite cartelle che conservano i programmi specifici della macchina, i file mftrace, potrace e FontForge; tutti e tre possono essere scaricati dalla rete; si possono trovare i luoghi da cui scaricare il tutto cercandoli con un motore di ricerca a piacere. Esistono versioni per ogni piattaforma, tranne che per le macchine Windows; su queste, però, si può installare un ambiente di simulazione di UNIX che si chiama cygwin, e all'interno di questo si possono installare i programmi suddetti come se la piattaforma Windows fosse una piattaforma UNIX.

Dalla finestra comandi, terminal, console, o xterm, che dir si voglia, si lancia il programma mftrace seguito dalle opzioni e dal nome del file da trasformare; per esempio

```
mftrace -pfb --simplify --potrace ocrb5
```

Se le variabili di environment, anche per cygwin, sono a posto, mftrace produce i font `.pfb` e, come sottoprodotto, i file `.tfm`; al massimo quel che resta da fare consiste nello spostare i file prodotti nelle opportune cartelle, eventualmente creandole apposta: `.../fonts/tfm/ocr/` e `.../fonts/type1/ocr/`.

Se si vuol sapere che cosa succede durante l'esecuzione di mftrace, si sappia che dapprima viene lanciato METAFONT sul file sorgente `ocrb5.mf` essendo stato specificato un fortissimo ingrandimento; il file di uscita di METAFONT viene convertito in un altro formato, i cui dettagli non ci interessano, e questo file viene passato a potrace. Questo determina i poligoni che descrivono al meglio i contorni delle lettere rasterizzate di cui è necessario determinare le spline di Bézier componenti; il lavoro viene ripetuto per ogni carattere nel font, e tutti

questi poligoni vengono utilizzati per determinare i nodi e i punti guida delle curve di Bézier osculatrici dei contorni da definire; con queste informazioni, e altre tratte dal file `.tfm` che era stato implicitamente generato da METAFONT nel primo passo, viene costruito un file temporaneo che viene poi inviato a FontForge con l'ordine di semplificare le curve che descrivono i contorni e di costruire il file `ocrb5.pfb`. Certo è più complicato da dire che da fare.

Creati i file `.pfb` di tutta la collezione si crea una semplice mappa; in questo caso che i font sono pochi si può procedere a mano; in condizioni diverse, con decine o centinaia di font, bisogna sfruttare le potenzialità del proprio sistema operativo. La mappa sarà dunque, nel nostro semplice caso:

```
ocrb5           <ocrb5.pfb
ocrb6           <ocrb6.pfb
ocrb7           <ocrb7.pfb
ocrb8           <ocrb8.pfb
ocrb9           <ocrb9.pfb
ocrb10          <ocrb10.pfb
```

e la si può chiamare `ocrb.map` mettendola in una cartella dove `updmap` la può trovare; in ogni caso bisogna aggiornare il database dei nomi dei file; si spostano i file `.pfb` appena generati un una nuova cartella dell'albero locale, per esempio in `.../fonts/type1/ocr/` e si aggiorna il solito database.

Bisogna ancora creare il file `.fd` per poter accedere a questi font da LaTeX e da pdfLaTeX. Detto file, molto semplice, appare così:

```
\ProvidesFile{uocr.fd}
                [2007/08/05 v1.0 Custom OCR font definitions]
\DeclareFontFamily{U}{ocr}{\hyphenchar\font-1}
\DeclareFontShape{U}{ocr}{m}{n}%
     {<5><6><7><8><9>gen*ocr%
     <10->ocr10}{}
\endinput
```

L'encoding di questo font è dichiarato 'unknown', non perché sia del tutto sconosciuto, ma perché esso non è completo come quello dei font che rispettano, per esempio, l'encoding OT1. Viene specificato il codice `-1` per la lineetta di 'a capo'; siccome nessun carattere ha l'indirizzo `-1`, questo significa che le parole scritte con questo font non possono essere divise in sillabe.

La famiglia viene definita con un nuovo nome, cosicché è necessario definire anche un comando per poter scegliere questa famiglia. È conveniente allora predisporre un piccolo file di estensione che contenga tutto il necessario:

```
\ProvidesPackage{ocr.sty}%
   [2007/08/05 v.1.0 Extension package for using OCR fonts]
\input{uocr.fd}
\DeclareRobustCommand{\ocrfamily}%
         {\fontencoding{U}\fontfamily{ocr}\selectfont}
\DeclareTextFontCommand{\textocr}{\ocrfamily}
\endinput
```

In questo modo la dichiarazione `\ocrfamily` dichiara di voler usare i font per lettura ottica, mentre il comando `\textocr` esegue la stessa operazione sul

suo argomento, provvedendo anche, se occorre, all'inserimento della correzione italica.

Si noti che nel file `.fd` i corpi da 5 pt a 9 pt sono ricavati ciascuno dal suo file specifico, mentre i corpi da 10 pt in su sono tutti ricavati per (eventuale) ingrandimento del font disegnato per il corpo di 10 pt. Per maggiori delucidazioni sulla manipolazione dei font e dei loro file di definizione il lettore è invitato a riferirsi al file `.../doc/latex/base/fntguide.dvi`.

Infine si lancia `updmap` come già descritto, e il proprio sistema, se non si sono commessi errori, è pronto per usare senza problemi i nuovi caratteri per lettura ottica.

11.7.1.3 Installare font commerciali

L'installazione di font commerciali, font Type1 o TrueType già presenti sul proprio calcolatore, corrisponde a fare operazioni simili a quelle viste nel paragrafo precedente, ma in un certo senso più facili.

I font Type1 sono disponibili insieme ai rispettivi file metrici con estensione `.afm`; i font TrueType non ne hanno bisogno, perché le informazioni metriche sono già contenute nel file `.ttf` che li descrive.

Ogni sistema TeX contiene già alcuni file eseguibili, fra i quali spiccano `afm2tfm` e `ttf2pt1`. Il primo programma accetta numerose opzioni, da rilevare dalla documentazione, e produce un file `.tfm` adatto al sistema TeX, partendo da un file in formato `.afm`; esso produce anche il file con il vettore di codifica `.enc` e il file virtuale `.vf` che consentono a ogni programma di composizione del sistema TeX di trattare il font come se fosse un normale font generato con METAFONT; nel momento in cui si tratta di generare un file di uscita in formato PostScript o PDF, gli eseguibili appositi sanno dove trovare le mappe e le informazioni di ricodifica per mettere i segni giusti nel file di uscita.

Il programma `ttf2pt1`, invece trasforma il font TrueType in un font Type1, generando anche il file `.afm`; a questo si può applicare la procedura del capoverso precedente.

Come al solito bisogna fare attenzione ad aggiornare sempre il database dei nomi dei file; se i programmi non lo fanno da soli, bisogna provvedere a creare la mappa specifica per ogni font trasformato e inserirne la riga apposita in un file `.map` da far leggere a `updmap` per generare i soliti tre file generali che verranno usati dai programmi destinati a produrre i file di uscita nel formato finale.

Talvolta questi programmi, specificando le opzioni giuste, sono in grado di produrre anche il file di descrizione `.fd` per definire le nuove famiglie, altrimenti è necessario provvedere a mano in modo autonomo.

11.7.2 Riassunto delle operazioni di gestione dei font e della loro connessione con l'intero sistema TeX

Vale la pena di riassumere graficamente il modo di operare del sistema TeX quando si usi LaTeX o pdfLaTeX. Molte delle informazioni sono sparse in diversi altri capitoli, ma qui, dove si parla di font, sembra il posto giusto per collegare tutto assieme.

Nelle figure 11.3–11.7 i simboli grafici hanno i significati seguenti:

Font I file che si riferiscono ai font in senso stretto sono rappresentati da cerchi gialli.

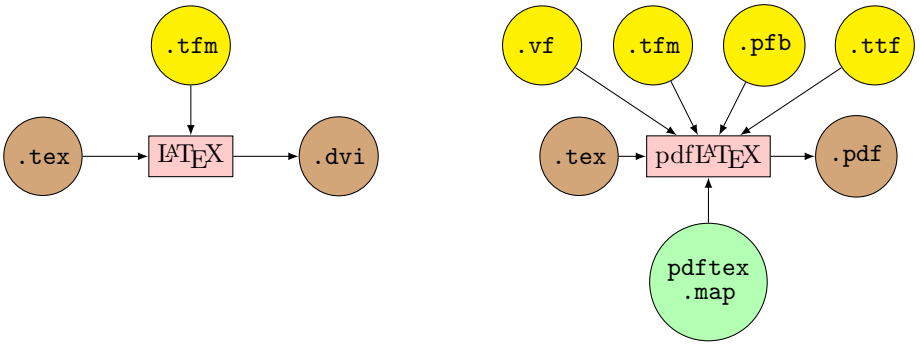

Figura 11.3: Schema grafico del processo di composizione

Programmi di sistema Sono rappresentati da rettangoli rosso chiaro.

Programmi esterni Sono rappresentati da rettangoli azzurri.

File accessori per la gestione dei font Sono rappresentati da cerchi verdi.

File finali Sono rappresentati da cerchi marroncino chiaro; 'file finale' indica il file che contiene un documento già composto.

Risorse esterne Sono gruppi di file di programmi e/o di font da reperire gratuitamente in rete, o onerosamente presso i rivenditori specifici. Sono rappresentati da ellissi blu chiare.

Uscite L'uscita su schermo, su carta, su lastre per fotolitotipografia, o su altri supporti sono rappresentati mediante ellissi arancione.

Nella figura 11.3 sono rappresentati i due processi di composizione descritti in questo testo, quello attraverso LaTeX e quello attraverso pdfLaTeX. Si noti la semplicità del processo attraverso LaTeX, a confronto con la relativa complicazione attraverso il processo pdfLaTeX. Il primo richiede solo il file sorgente e il file metrico dei font usati, e in uscita presenta il risultato 'finale' mediante un file `.dvi`. Questo file è finale solo in senso logico, perché, comunque, per essere utilizzato ha ancora bisogno di trasformazioni, specialmente se vi sono state incluse delle immagini ottenute da file esterni.

Al contrario il processo di composizione mediante pdfLaTeX è completo e il file finale si può considerare definitivo; per fare ciò, però, esso ha bisogno di tutti i file concernenti i font, specialmente quelli di tipo outline.

Il file `.dvi` può essere visualizzato e/o stampato direttamente mediante software che solitamente fa già parte del corredo del sistema TeX; sarà il visualizzatore YAP di MiKTeX, oppure xdvi di Linux oppure altri programmi simili. Lo stesso può dirsi del file `.pdf` in uscita da pdfLaTeX, così come del file `.ps` in uscita dal convertitore dvips, nel caso che sia richiesto questo formato per l'uso successivo. Nella figura 11.4 sono rappresentati graficamente i processi di presentazione mediante i vari programmi specifici e sono messi in evidenza i file di cui necessitano, con particolare rilievo per quelli che riguardano i font.

Generalmente però il file `.dvi` ha bisogno di essere trasformato o in file `.pdf` o in file `.ps`; quest'ultimo a sua volta può richiedere la trasformazione

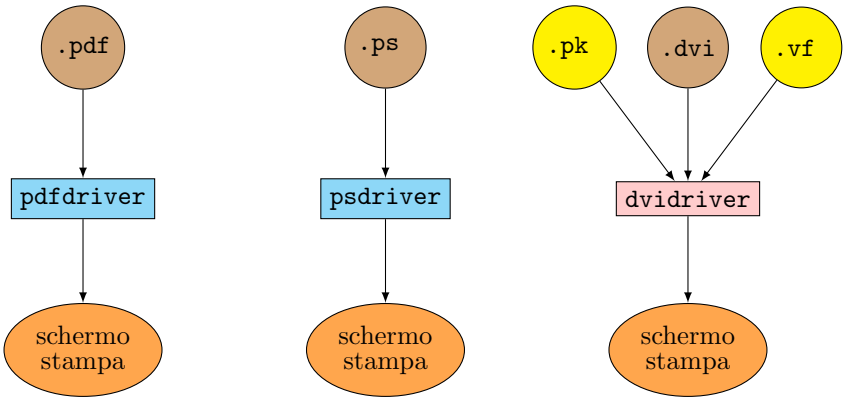

Figura 11.4: Presentazione all'esterno dei risultati della composizione

ulteriore in file `.pdf`; il processo di conversione è rappresentato graficamente nella figura 11.5, ancora evidenziando l'importanza dei file concernenti i font.

Come si vede da queste figure, i file direttamente o indirettamente legati ai font sono tanti e spesso bisogna crearli apposta con il supporto delle utility fornite con il sistema TeX o con l'ausilio di programmi esterni. La figura 11.6 mostra in forma schematica le risorse esterne da cui trarre i font necessari e come trasformare questi ultimi in modo da renderli usabili con il sistema TeX. Si vede chiaramente che i font in formato METAFONT sono i più facili da usare, il che è logico, visto che TeX e METAFONT sono programmi 'fratelli' entrambi concepiti da Donald Knuth e sono sempre distribuiti assieme e completi di una buona collezione di file sorgente di tipo `.mf`. Ma i font outline distribuiti secondo lo standard Adobe (PostScript Type1), oppure i font TrueType, nati da un progetto comune della Adobe e dalla Microsoft, richiedono ulteriori trattamenti, se non altro perché non sono dotati dei file `.tfm`, che bisogna quindi generare, ma spesso hanno anche codifiche diverse, certamente diverse da quelle che TeX si aspetta, e quindi sono necessarie conversioni a diversi livelli. I programmi che operano queste conversioni e il loro modo di procedere sono illustrati schematicamente nella figura 11.6 in questa figura per altro è indicata anche la conversione dei font `.mf` in font di tipo outline, che sono così utili specialmente per la visualizzazione su schermo, dove è necessario poter ingrandire o rimpicciolire la schermata, e quindi il suo contenuto, secondo le necessità di lettura.

In particolare le mappe parziali dei font disponibili sulla macchina particolare sulla quale viene eseguita la composizione vanno unite in modo coerente fino a formare le mappe generali di cui necessitano i programmi di conversione dei file finali. Il procedimento è indicato schematicamente nella figura 11.7; il programma indicato con il nome 'editor' è un qualunque programma di gestione di file testuali ASCII; potrà essere notepad dei sistemi Windows, oppure vim dei sistemi Linux, ma va benissimo anche lo shell editor usato per gestire i file sorgente da comporre con il sistema TeX.

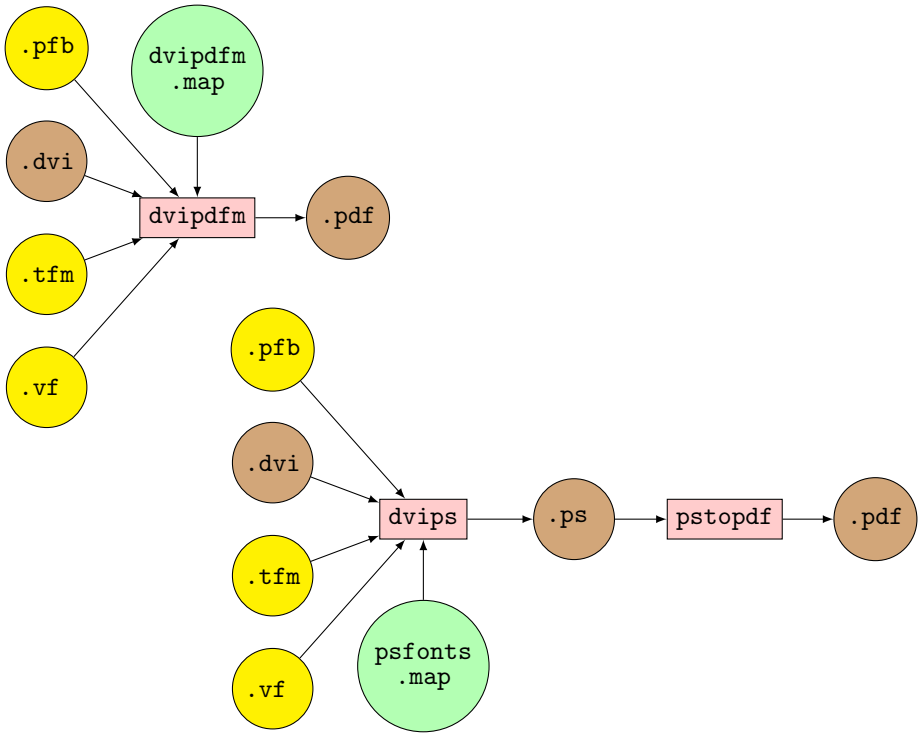

Figura 11.5: Processi di conversione dei vari formati di uscita dai programmi di composizione

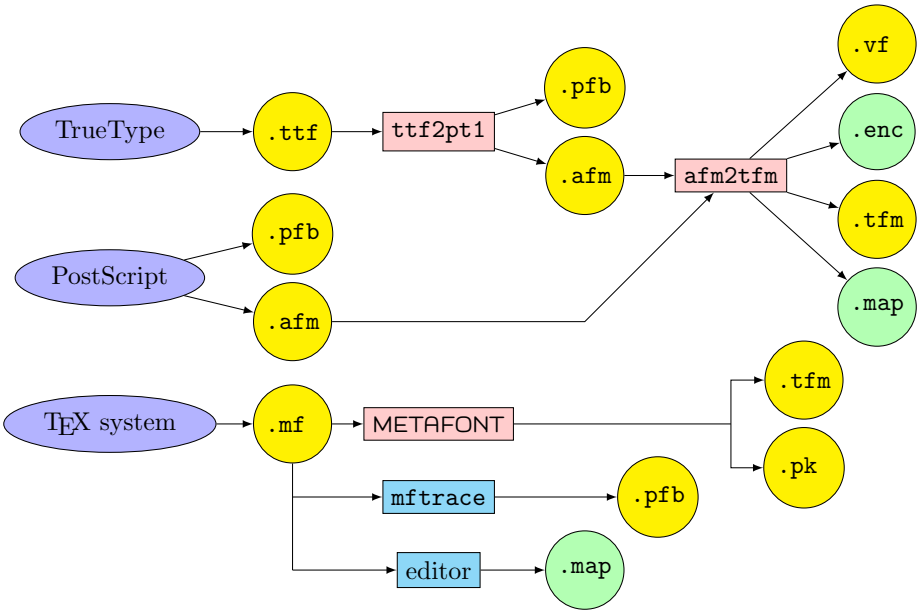

Figura 11.6: Trasformazioni di formato dei font e generazione dei file ausiliari

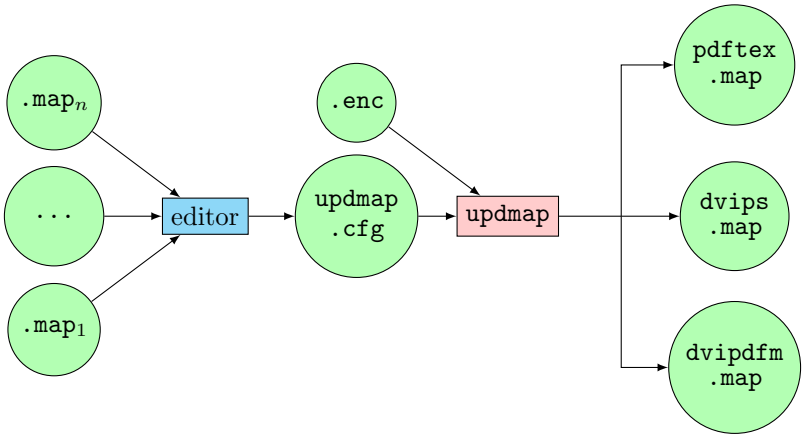

Figura 11.7: Gestione delle mappe dei font

11.8 Conclusioni

Tutte queste operazioni, descritte forse con troppo dettaglio, possono impaurire il neofita; tuttavia ci si può consolare pensando che l'installazione di un nuovo font non è una operazione da eseguirsi tre volte al giorno; d'altra parte se così fosse, queste operazioni finirebbero per venire eseguite in modo del tutto automatico e non apparirebbero così intricate.

Capitolo 12

LaTeX: le presentazioni

12.1 Introduzione

Non è possibile tralasciare, nemmeno in una guida introduttiva, l'argomento delle presentazioni.

Per 'presentazioni' si intendono quelle proiezioni che accompagnano una conferenza, un discorso pubblico, un seminario, una lezione, eccetera.

Se il contenuto delle schermate da proiettare è prevalentemente o completamente testuale, al massimo include delle figure, vanno benissimo i programmi liberi oppure commerciali, i quali spesso producono proiezioni molto accattivanti, ricche di effetti speciali e piene di parti colorate; spessissimo anche lo sfondo della schermata contiene delle parti disegnate, o delle fotografie, o il logo dell'istituzione di appartenenza dell'oratore; perché non disturbino la lettura queste immagini di sfondo dovrebbero essere molto chiare o comunque alterare il meno possibile il colore dominante dello sfondo.

Tuttavia preparare delle belle presentazioni è un'arte; se poi la presentazione deve contenere dello scritto tecnico (formule matematiche, schemi filologici, o altre cose non facenti parte della prosa 'normale') allora è opportuno usare LaTeX, il che comporta la necessità di saper usare LaTeX al meglio delle sue potenzialità; bisogna conoscere l'arte di comporre con LaTeX oltre che l'arte di predisporre delle schermate efficaci per il trasferimento sintetico dell'informazione dall'oratore all'ascoltatore.

12.2 Le classi per le presentazioni

LaTeX nasce con una semplicissima classe per le presentazioni, `slides`; nonostante la sua semplicità, estensibile facilmente con i vari pacchetti di estensione descritti anche negli altri capitoli, essa contiene alcuni elementi importanti da tenere presente.

1. Poiché l'ascoltatore recepisce e ricorda meglio quello che legge, che sente, e che vede sullo schermo, è importante che ogni slide, o schermata, contenga un solo concetto e che l'ascoltatore non sia sottoposto a più di un nuovo concetto al minuto.

In realtà un concetto può essere svolto anche su due o tre slide, ma è preferibile essere sintetici e non è il caso di sommergere l'ascoltatore sotto una valanga di slide, ognuna necessariamente proiettata per poco tempo. Per questo scopo è quindi necessario essere molto sintetici.

2. I due elementi illustrati sopra implicano che ogni slide non deve contenere più di una dozzina di righe di testo (e 12 forse sono già tante) contando anche le righe occupate da espressioni matematiche o da testi in alfabeti speciali o contenenti segni speciali; ognuna di queste righe va forse contata per due!

3. Perciò se la slide viene stampata su fogli di metacrilato (i *lucidi*, chiamati anche impropriamente 'trasparenze') allora i font 'normali' devono essere oltre che facilmente leggibili anche di corpo molto grande. La classe `slides` usa di default per il corpo normale il corpo da 20 punti (per l'esattezza 19,907 pt) della famiglia senza grazie; per migliorare la leggibilità questa famiglia in realtà è una famiglia speciale con l'altezza delle minuscole maggiore del solito: abcdefg; l'esempio ha lo stesso corpo usato nel resto del testo, ma come si vede le sue minuscole sembrano decisamente più grandi; in realtà è più grande la *x-height* ma sono più corti gli ascendenti.

4. Tenendo conto che lo schermo di proiezione ha solitamente il rapporto 4 : 3 fra base ed altezza, anche il foglio di metacrilato o la slide viene stampata facendo riferimento alla base come il lato lungo del foglio e bisogna usare nella dichiarazione di classe l'opzione *landscape*; in questo modo aumenta un poco la giustezza, ma diminuisce l'altezza e il numero di righe disponibile rimane necessariamente basso; non è possibile, perciò mettere troppa informazione in ciascuna slide.

5. La classe `slides` non consente di eseguire animazioni, specialmente si si fa uso dei lucidi di metacrilato; altre classi lo consentono se e solo se non si usano i lucidi, ma si proietta direttamente dal PC o dal laptop mediante un videoproiettore.

   Però con l'uso del pacchetto **color** è possibile colorare lo sfondo e scegliere colori diversi di ogni parte della slide composta con LaTeX, sia testo, sia matematica, sia disegni eseguiti con l'ambiente *picture* o altri ambienti nativi del sistema TeX.

12.3 Altre classi per le presentazioni

Ogni distribuzione del sistema TeX contiene diverse classi per predisporre ottime presentazioni; si possono citare i pacchetti **prosper** e il suo successore **powerdot**, **seminar**, **pdfscreen**, **pdfslide**, **ppower4**, **texpower**, per finire con **beamer**.

Ognuno di questi pacchetti produce slide bellissime e le presentazioni di dimostrazione, che accompagnano ogni pacchetto fra la sua documentazione, sono tutte accattivanti.

Ogni pacchetto offre una o più classi per la predisposizione di vari tipi di presentazione; ognuno finisce poi per affezionarsi ad un particolare pacchetto, ma le possibilità di personalizzazione sono tali e tante che non si corre il rischio di eseguire presentazioni con lo stesso stile standard.

12.4 La classe beamer

In inglese *beamer* significa videoproiettore. Il pacchetto **beamer** mette a disposizione del compositore sia la classe `beamer` sia una serie di file di estensione modulari con i quali è possibile personalizzare le proprie presentazioni e le modalità per attribuire valori diversi ai diversi parametri che caratterizzano sia il layout delle slide, sia i tipi di comandi o di segni particolari che appaiono nelle slide.

Queste sono concepite con file PDF interattivi, per cui ogni slide contiene anche, disegnati in colore molto tenue, tanto da non disturbare la lettura, anche i 'pulsanti' per muoversi nella sequenza di slide che costituiscono la presentazione. Inoltre è possibile avere a disposizione una specie di indice generale della presentazione sempre visibile, così che gli ascoltatori possano seguire il cambiamento di colore del titoli delle 'sezioni' che costituiscono la presentazione; questi stessi titoli costituiscono degli *hyperlink* utili all'oratore per navigare rapidamente attraverso la propria presentazione senza avere la necessità di far scorrere la presentazione fino a trovare la slide giusta.

La classe prevede l'uso dei pacchetti **pgf** e **xcolor** così che si possano sfruttare le molteplici funzionalità del pacchetto **pgf** per quel che riguarda la possibilità di eseguire dei bellissimi disegni o per inserire le funzionalità di quel pacchetto per l'inserimento delle immagini esterne.

Invece il pacchetto **xcolor** consente di gestire molto semplicemente i colori per ottenere sia colori misti, sia gradienti di colore lineari o sferici, il tutto con comandi di alto livello molto semplici per il compositore.

Le slide possono avere il loro contenuto esposto incrementalmente; vale a dire che, per esempio, la dimostrazione di un teorema può essere fatta esponendo una dopo l'altra le frasi che ne costituiscono la sequenza logica, senza che il testo già esposto cambi posizione via via che nuovo testo viene esposto. Uno dei vantaggi per l'oratore è che quanto verrà esposto in slide successive appare scritto in colore chiarissimo anche sulle slide precedenti, così che l'ascoltatore quasi non se ne accorge, ma l'oratore ne trae un notevole aiuto perché sa già in anticipo, vedendola, quale sarà la slide successiva.

Ovviamente la classe `beamer` consente di eseguire delle transizioni fra una slide e l'altra agevolando l'ascoltatore nel capire sia i collegamenti fra una slide e la successiva, sia, con transizioni diverse, nel capire quando si passa da una sezione all'altra della presentazione.

La predisposizione delle slide è piuttosto semplice: ognuna corrisponde ad un ambiente *frame*; ogni frame accetta un titolo e il testo della slide; se la slide deve essere presentata in fasi successive, il file di uscita conterrà diverse 'pagine', ma il frame resta invariato; semplicemente i comandi per l'esposizione incrementale delle varie parti della slide fanno parte, nel file sorgente, del testo contenuto nel frame.

Non è possibile in un testo stampato presentare questi effetti dinamici; ma si invita il lettore ad esaminare con attenzione uno del file 'demo' che accompagna il pacchetto beamer; si trovano tutti in `.../doc/latex/beamer/examples/`; suggerisco `beamerexample5.pdf`, ma certi file sono solo per presentare alcuni dettagli, altri sono per presentazioni complete. Un altro esempio è formato dalla coppia di file `beamerexample2.article.pdf` e `beamerexample2.beamer.pdf` derivati dal medesimo file sorgente; a seconda di quali parametri vengono stabiliti nel preambolo, dallo stesso sorgente si possono ricavare sia la presentazione sia

un articolo che può costituire direttamente il materiale 'cartaceo' da distribuire all'uditorio.

12.5 La documentazione

Tutti i pacchetti nominati dispongono della loro propria documentazione sul percorso `.../doc/latex/`⟨*pacchetto*⟩`/`. Il file di documentazione di **beamer** si chiama `beameruserguide.pdf` ed è particolarmente interessante, non solo perché contiene una documentazione molto ben fatta, tenendo conto anche della versatilità del pacchetto, e quindi della moltitudine di comandi e ambienti che esso fornisce, ma anche per le preziose informazioni sulla preparazione di presentazioni professionali; questo genere di informazioni non è facilmente reperibile in altra documentazione o in altri libri.

La sostanza è che quasi tutto quello che è opportuno fare da un punto di vista tecnico-compositivo è disponibile con LaTeX e con i comandi messi a disposizione dai pacchetti di servizio **pgf** e **xcolor**, ma quello che deriva dalla professionalità e dal buon gusto non può e non è solitamente descritto da nessuna parte, tranne, appunto, nel manuale di **beamer**.

È sorprendente notare come la maggior parte delle presentazioni composte con noti programmi dedicati, sia commerciali sia freeware, pur essendo quei programmi del tutto validi per lo scopo, peccano perché gli autori non conoscevano e non avevano nessun modo di ricavare dalla documentazione le regole di stile e di buon gusto per la predisposizione di slide efficaci; queste sono efficaci se sono prive di quegli elementi 'decorativi' che, invece di favorire la trasmissione della comunicazione, distraggono lo spettatore/ascoltatore, il quale alla fine della conferenza o della lezione si ricorda di queste animazioni ma non si ricorda di quello che ha detto l'oratore.

È quindi assai raccomandabile leggersi attentamente il manuale di **beamer**; si impareranno moltissime cose che permetteranno di usare lo strumento compositivo al meglio per ottenere lo scopo di una presentazione, cioè quello di trasmettere un messaggio all'ascoltatore nel modo più efficace possibile.

12.6 Una breve presentazione

Nella figura 12.1 sono rappresentate 8 slide corrispondenti a 5 frame di una brevissima presentazione. Esse sono disposte in verticale; la prima colonna contiene le slide da 1 a 4 e la seconda colonna da 5 a 8.

Il codice per produrre queste slide è il seguente e serve per interpretare il risultato ottenuto nelle slide rappresentate nella figura 12.1.

```
%%% file per produrre alcune slide per GuidaGuit
\documentclass{beamer}
%                                   Preambolo
\usetheme{AnnArbor}
\useoutertheme[right]{sidebar}
\setbeamercolor{alerted text}{fg=red!90!black}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
\usepackage{pgf}
```

Figura 12.1: Otto slide per una presentazione di cinque frame

```
\usepackage{pict2e,curve2e}
\usepackage[color]{guit}
\usepackage[italian]{babel}

\beamertemplatetransparentcovereddynamic

\title{I numeri complessi}
\subtitle{Conferenza internazionale
del~Gruppo~Italiano~degli~utenti~di~\TeX}
\author{Mario Rossi}
\institute{\GuIT}

\date{Pisa, 20--22 ottobre 2015}


\pgfdeclareimage[width=\textwidth]{CifreIndiane}%
                                {CifreIndianeVIIsecolo}
\pgfdeclareimage[width=15.5mm]{guitlogo}{GuITlogo}

\logo{\pgfuseimage{guitlogo}}
% Fine del preambolo

\begin{document}% Inizio della presentazione


\begin{frame}% Primo frame, prima slide
\titlepage
\end{frame}

\begin{frame}% Secondo frame, seconda slide
\frametitle{Indice}
\tableofcontents
\end{frame}

\section{Introduzione storica}

\begin{frame}% Terzo frame, terza slide
\frametitle{Una breve storia dei numeri}
All'inizio del \textsc{vii} secolo gli Indiani inventarono
la notazione posizionale e le <<nove>> cifre dall'1 al 9;
lo zero veniva detto a voce ma non aveva ancora un suo segno.

\begin{center}
\pgfuseimage{CifreIndiane}
\end{center}
\end{frame}

\section{Nascita dei numeri complessi}
```

```
\begin{frame}% Quarto frame,  diviso in quattro slide
\frametitle{I numeri complessi nascono nel '500}
\begin{enumerate}
\item<1-> Nel XVI secolo Tartaglia e Cardano introducono
la radice quadrata di $-1$
\item<2-> Il nome di \emph{unità immaginaria} viene creato
da Réné Descartes nel 1637
\item<3-> Gauss nel 1799 contribuisce con i suoi scritti
a diffondere i numeri complessi
\item<4-> Hamilton nel 1833 pubblica la teoria dei numeri
complessi
\end{enumerate}
\end{frame}


\section{I numeri complessi in \texttt{pict2e}}

\begin{frame}% Quinto frame, ottava slide
\frametitle{I numeri complessi come operatori geometrici}
\begin{center}\unitlength=1mm
   \begin{picture}(60,40)
   \put(0,0){\vector(1,0){60}}\put(60,1){\makebox(0,0)[br]{$x$}}
   \put(0,0){\vector(0,1){45}}\put(1,45){\makebox(0,0)[lt]{$y$}}
   \thicklines
   \put(0,0){\textcolor{red}{\vector(3.4641,2){40}}}
   \thinlines
   \multiput(40,-.5)(0,2){12}{\line(0,1){1}}
   \multiput(-.5,23.094)(2,0){20}{\line(1,0){1}}
   \put(41,1){\makebox(0,0)[bl]{$a$}}
   \put(1,24.1){\makebox(0,0)[bl]{$b$}}
   \VectorArc(0,0)(20,0){30}
   \put(20,4){\makebox(0,0)[bl]{$\varphi$}}
   \put(20,12.6){\rotatebox{30}{\makebox(0,0)[b]{$m$}}}
   \put(0,0){\thicklines\textcolor{blue}{\vector(1,0){10}}}
   \put(5,-1){\textcolor{blue}{\makebox(0,0)[t]{vettore
              unitario}}}
   \end{picture}
\end{center}
Visto come operatore geometrico, il numero complesso
$m\mathsf{e}^{\mathsf{i}\varphi}=a+\mathsf{i}b$ agisce su
un vettore; in questa figura agisce sul
\textcolor{blue}{vettore unitario blu}; lo scala tramite
il fattore \alert{$m$} e lo ruota dell'angolo \alert{$\varphi$}
producendo il \alert{vettore rosso}.
\end{frame}

\end{document}
```

Nel preambolo appaiono diversi comandi, alcuni dei quali sono già familiari, per invocare **babel** con l'opzione *italian* al fine di comporre in italiano; per invo-

care **inputenc** con l'opzione *latin1* per scrivere usando le lettere accentate; per invocare **outputenc** con l'opzione *T1* per scrivere nel file di uscita con le lettere accentate direttamente, non ottenute per sovrapposizione del segno dell'accento sopra il segno della vocale. Si invoca anche il pacchetto **pgf** per poter usufruire della grafica avanzata, e per la gestione delle figure; il pacchetto **guit** per gestire i loghi del Gruppo degli utilizzatori Italiani di TeX; i pacchetti **pict2e** e **curve2e** per disegnare in uno dei frame della presentazione.

Ma, ai fini di questo esempio, sono interessanti gli altri comandi:

`\usetheme` serve per scegliere il layout generale dei frame, sia come colori, sia come disposizione delle informazioni che corredano la presentazione; in questo caso si è scelto lo stile `AnnArbor` che è uno dei vari stili messi a disposizione dell'utente di **beamer**; ovviamente questi sono predefiniti, ma l'utente può crearsi dei temi a suo piacimento partendo da zero, oppure può copiare e modificare quelli esistenti mediante gli altri comandi che si vedranno qui di seguito.

`\useoutertheme` serve per scegliere le parti che contornano i frame ai fianchi; in questo caso si è scelto di usare la barra laterale (una striscia con un colore di fondo nella quale appariranno certe informazioni) e si è scelto di mantenerla a destra.

`\setbeamercolor` serve per impostare uno dei colori che beamer usa per certi tipi di testo; in questo caso il colore dell'*allerted text*, il testo su cui richiamare l'attenzione, viene impostato con il 90% di rosso e il restante (10%) nero; insomma si preferisce un rosso un po' più scuro del normale.

`\beamertemplatetransparentcovereddynamic` è una dichiarazione con la quale si chiede che il testo da esporre in slide successive sia 'trasparente', molto chiaro in modo da intravvederlo anche quando il testo non è esposto.

`\title` `\author` e `\date` sono vecchie conoscenze, ma **beamer** consente di specificare anche un sottotitolo per la presentazione mediante il comando `\subtitle`. Il modo particolare con cui è scritto il sottotitolo serve per garantire che la frase sia divisa in due sottofrasi in modo che ognuna contenga le sue congiunzioni e preposizioni semplici o articolate.

`\institution` serve per inserire il nome o il logo dell'istituzione a cui appartiene l'oratore.

`\pgfdeclareimage` è un comando che consente di assegnare non solo certe caratteristiche all'immagine da usare, in questo caso la larghezza, ma anche un nome con cui richiamarle insieme alle loro proprietà. Attenzione, non si tratta di una cosa da poco; l'immagine `GuITlogo`[1] viene riprodotta in ogni slide, quindi se ogni volta si ricaricasse il file del disegno, si correrebbe il rischio di finire con file dalla dimensioni enormemente grandi; usando invece `\pgfdeclareimage` il particolare file che contiene l'immagine viene caricato una sola volta grazie al fatto che il programma pdfLaTeX è in grado di riutilizzare quell'immagine un numero illimitato di volte senza

[1] Qui si è creata apposta questa immagine, trasformando il contenuto di ciò che viene prodotto con il comando `\GuITmeeting`; il logo del gruppo GuIT non mi è stato possibile scaricarlo dalla rete.

copiarla ma facendovi riferimento mediante i suoi meccanismi interni di gestione degli hyperlink.

`\logo` dice quali comandi devono venire usati per produrre l'immagine del logo nel punto in cui questo deve apparire.

`\pgfuseimage` infine è il comando che, riferendosi al nome di un'immagine precedentemente dichiarato mediante `\pgfdeclareimage`, effettivamente crea gli hyperlink interni che rendono visibile l'immagine nel punto voluto.

Con l'inizio del documento appaiono le dichiarazioni dei vari frame mediante i successivi ambienti *frame*. Il primo frame contiene il titolo della presentazione; il comando `\titlepage` fa sì che il frame sia composto in modo un po' differente dagli altri, ma che contenga buona parte delle informazioni relative al titolo, al presentatore, e alla sua istituzione di appartenenza, all'evento nell'ambito del quale si svolge la presentazione, eccetera; la prima slide e primo frame della figura 12.1 mostra tutti questi dettagli. Nell'angolo in basso a destra compare anche il numero progressivo del frame riferito al numero complessivo dei frame che compongono la presentazione; nell'esempio compare la frazione 1/5 per dire che si tratta del primo di cinque frame.

Il secondo frame contiene l'indice della presentazione; è utile che questo indice venga presentato all'inizio, perché è un semplice mezzo per consentire all'oratore di delineare all'uditorio il piano della sua presentazione. Questo stesso elenco è collegato internamente agli hyperlink che consentono di navigare all'interno della presentazione; le stesse voci di questo indice compaiono anche nella barra laterale destra, e si illuminano di un colore più chiaro ogni volta che si entra ad esporre le varie slide che compongono una data sezione; gli ascoltatori seguendo questa barra possono rendersi conto istante per istante del punto a cui è arrivato l'oratore, senza bisogno di controllare l'orologio con mosse ampie e vistose per farsi notare dall'oratore affinché ci dia un taglio... Spero che nessuno dei lettori di queste note si trovi mai a fare o a subire queste azioni!

Tra il secondo frame e il terzo è inserito un comando di sezionamento; il suo contenuto consente di creare l'indice, ma non compare mai al di fuori dell'indice; è bene che una presentazione sia strutturata in sezioni, eventualmente anche in sottosezioni, ma non si frammenti troppo l'esposizione; durante un congresso è possibile che ogni oratore abbia a sua disposizione 20 minuti di cui cinque sono destinati alle risposte alle domando poste dagli ascoltatori; quindi la presentazione non dovrebbe contenere più di 15–20 slide; due o tre sezioni sono più che sufficienti per strutturare la presentazione. Per una lezione, magari di due ore, è evidente che si può strutturare maggiormente, ma anche in questo caso non è il caso di fare un unico file con un indice interminabile; gli allievi si demoralizzerebbero, specialmente tenendo conto che durante una lezione assistita da una presentazione, quanto viene detto è molto di più di ciò che si potrebbe esporre con le sole parole o con l'aiuto di una lavagna su cui scrivere o disegnare. Una simile lunga lezione è meglio che sia strutturata in due parti (due distinti file) da presentare con un congruo intervallo fra le due.

Si vede che nel terzo frame è richiamata per nome una delle due immagini precedentemente dichiarate e si fa uso di `\pgfuseimage` che svolge il ruolo che normalmente avrebbe il comando `\includegraphics` del pacchetto **graphicx**.

Il quarto frame corrisponde a 4 slide; si vede che esso contiene una enumerazione, ma dopo ogni comando `\item` compare una strana espressione: due

parentesi 'acute' racchiudono un intervallo nel quale sono indicate le slide da proiettare; il primo termine dell'elencazione deve essere proiettato dalla slide 1 in poi; il secondo elemento dalla slide 2 in poi, eccetera; la sintassi di questo intervallo di numerazione delle slide segue questa regola: se le parentesi acute contengono un solo numero, senza il trattino, la slide viene proiettata solo quando tocca a lei (in base al numero d'ordine); se invece l'espressione contiene un intervallo del tipo `2-3` il termine dell'elencazione appare solo per le slide da 2 a 3 comprese, ma non compare mentre viene proiettata la slide numero 1 né quando viene proiettata la numero 4. Se manca un numero prima del trattino, si intende il numero 1; se manca un numero dopo il trattino, si intende fino all'ultima slide di questo frame. Il fatto, però, è che le slide dello stesso frame hanno tutte lo stesso contenuto; ma quelle che devono mostrare solo una parte del contenuto, ce l'hanno in colore trasparente molto chiaro, quasi invisibile per chi non sa che cosa c'è scritto, ma non per l'oratore che invece intravede quanto apparirà nella slide successiva.

Infine nel quinto frame c'è un disegno eseguito con i comandi dell'ambiente *picture*, come ridefinito dal pacchetto **pict2e** ed esteso con i comandi definiti nel pacchetto **curve2e**.

12.7 Osservazioni

Questo capitolo serve per richiamare l'attenzione del lettore sull'opportunità di documentarsi a fondo nelle rispettive guide in merito alle potenzialità dei vari pacchetti citati. Alcuni, come **beamer**, sono nati per produrre il risultato finale direttamente in formato PDF, altri ricorrono al pacchetto **PSTricks** e quindi devono subire due trasformazioni: prima devono essere composti con LaTeX per ottenere il file DVI, poi questo deve essere trasformato in formato PDF, per esempio con il programma dvipdfm oppure attraverso il doppio passaggio dvips e ps2pdf. Il gioco può valere la candela, come si dice, ma dipende dal tipo di disegni che è necessario inserire nelle slide.

È inevitabile che ognuno si affezioni ad una particolare classe, anche perché bisogna usare molti nuovi comandi dai nomi talvolta non così facili da ricordare; perciò l'abitudine diventa un fattore importante.

Però parlando di arte di comporre con LaTeX, non si può negare che LaTeX è uno strumento formidabile, ma da solo non è sufficiente per predisporre presentazioni efficaci e accattivanti. La necessità di documentarsi nell'arte della comunicazione orale e visiva diventa essenziale; padroneggiando bene quest'arte diventa poi abbastanza facile usare lo strumento LaTeX con i suoi pacchetti di estensione in modo da ottenere il meglio.

Capitolo 13

LaTeX: i vari tipi di documenti e stili di composizione

13.1 Introduzione

I file di classe, che servono per definire i parametri compositivi del particolare documento che si intende comporre, sono il cuore di LaTeX e sono proprio loro che interpretano il mark-up di LaTeX per eseguire la composizione in accordo con questo mark-up.

13.2 Classi standard

LaTeX viene distribuito con una piccola dotazione di file di classe abbastanza generici da poter affrontare la composizione di qualunque documento; tuttavia questa genericità ha favorito lo sviluppo di classi particolari di cui si dirà brevemente nei prossimi paragrafi.

book è la classe con cui è stato composto questo testo. Essa prevede di usare di default tutte le strutture di sezionamento, da `\chapter` fino alla più minuta suddivisione del testo costituita da `\subparagraph`; prevede la composizione sul recto e sul verso dei fogli, prevede la pagina contenente il titolo come pagina a se stante; il disegno grafico prevede uno spostamento della gabbia verso il centro; le testatine contengono il titolo corrente e il numero della pagina; il titolo corrente è scritto in caratteri maiuscoli inclinati; il documento può essere suddiviso in materiale iniziale, corpo del testo e materiale finale, con caratteristiche compositive adeguatamente diverse, eccetera.

report è la classe con cui si scrivono i rapporti 'tecnici': testi relativamente brevi, di solito non superiori ad una cinquantina di pagine, con il titolo facente parte della prima pagina dove comincia anche il testo; di default sono composti sul recto delle pagine, per cui la gabbia del testo è centrata. Non prevede la definizione dei comandi `\frontmatter`, `\mainmater`

e `\backmatter` perché un rapporto non è così strutturato come un libro, ma per il resto la classe assomiglia abbastanza alla classe *`book`*.

`article` è la classe con cui vengono composti brevi articoli, di solito di lunghezza non superiore alla decina di pagine, con il titolo sulla stessa pagina dove comincia il testo; spesso sono scritti su due colonne; hanno esplicitamente l'ambiente *`abstract`* per redigere un breve sunto del contenuto; non è definito il comando `\chapter` anche se continua ad essere definito il comando `\part`; nessuno vieta di dividere un articolo in parti, ma la struttura di sezionamento più alta è `\section`; spesso gli articoli sono senza indice generale, né indice analitico, ma dispongono quasi sempre di una bibliografia.

`letter` serve per comporre lettere commerciali; lo stile è molto americano, ma non è così difficile personalizzarlo per conformarsi allo stile italiano delle lettere commerciali; consente, volendo, di scrivere lettere con il medesimo testo ad una moltitudine di destinatari, così come consente di scrivere diverse lettere a destinatari diversi inserendole nello stesso file sorgente.

`ltnews` serve per comporre una semplice newsletter di solito di una sola pagina, composta su due colonne.

`ltxdoc` serve per comporre la documentazione dei file sorgente dei pacchetti che vengono distribuiti; questi file sorgente contengono simultaneamente la documentazione e il codice e possono servire per produrre diverse classi e/o file di estensione; il loro formato è molto particolare, ma sono preziosi per i programmatori per poter scrivere dei programmi in linguaggio TEX o LATEX in una forma ben documentata, senza che questi commenti disturbino il lavoro del motore TEX quando deve comporre un documento.

`ltxguide` serve per comporre le guide di documentazione dei vari aspetti di LATEX; si tratta delle guide incluse in ogni distribuzione e si trovano generalmente nella cartella `.../doc/latex/base/`; queste guide sono la fonte principale di informazione che l'utente dovrebbe consultare e 'conoscere a memoria'...

`minimal` è una classe minima, fatta apposta per avere una classe che garantisca il minimo di performance, essenzialmente pensata per collaudare la definizione di nuovi comandi o lo sviluppo e le prestazioni di nuovi pacchetti di estensione quando il programmatore li sta costruendo.

`proc` è una classe simile ad *`article`* che serve per produrre degli articoli secondo uno stile da *proceedings* (atti di una conferenza).

`slides` serve per predisporre dei lucidi di presentazione di un discorso o di una conferenza. In realtà una volta si usavano le lavagne luminose; oggi si usano i videoproiettori; con questi ultimi non si producono lucidi ma si proietta direttamente dal calcolatore; le prestazioni di *`slides`* sono adatte proprio ai lucidi su fogli di metacrilato. Per le presentazioni oggi si possono usare diversi programmi, ma, restando in ambito LATEX, esistono delle classi, come *`beamer`*, ottenuta con il pacchetto **beamer**, che producono delle presentazioni a colori bellissime, con animazioni finalizzate alla

presentazione, e con la qualità della composizione della matematica e della gestione/importazione di immagini efficaci come quelle che LaTeX offre.

13.3 La creazione di nuove classi

L'utente può crearsi nuove classi con due approcci (in verità non mutuamente esclusivi) molto semplici; (*a*) modificare le prestazioni di una classe esistente e (*b*) creare una classe ex novo.

Il primo approccio è decisamente il più semplice ed è affrontabile da chiunque; il file `clsguide` che si trova nella solita cartella `.../doc/latex/base/` contiene le indicazioni per seguire questa via e presenta non pochi esempi.

Il secondo approccio è decisamente molto più complesso, ma consente una personalizzazione maggiore.

Una via di mezzo è quella di utilizzare classi non standard ma fortemente personalizzabili come si vedrà nel prossimo paragrafo.

Infine, la via maestra è quella di non creare nessuna classe nuova, a meno che non sia fortemente indispensabile. Esistono infiniti pacchetti di estensione per modificare le prestazioni delle classi esistenti; qui se ne citeranno solamente alcuni, quelli più usati e considerati più utili.

fancyhead serve per personalizzare le testatine ed i piedini; di ognuno si può personalizzare quello che compare nella pagina di sinistra, invece che nella pagina di destra; di ognuno si può specificare che cosa scrivere nel centro, a sinistra, oppure a destra, così da avere sei posizioni in ogni pagina per inserire informazioni che, in definitiva, aiutano la navigazione nel testo, e quindi sono particolarmente indicate nei documenti molto strutturati. Mentre nelle testatine 'ordinarie', per esempio della classe *book*, le testatine, oltre al numero della pagina, possono contenere solo il titolo (corto) del capitolo o del paragrafo correnti, con **fancyhead** si riesce anche a inserire informazioni di terzo livello, per esempio il titolo dei sottoparagrafi.

caption e **ccaption** servono per personalizzare la composizione delle didascalie.

geometry serve per personalizzare il layout della pagina, la sua geometria, secondo schemi tradizionali o moderni, sempre parametrizzati, in modo da essere completamente liberi nella definizione dei margini, dei contrografismi fra testatina o piedino e il corpo della pagina; per scegliere la giustezza in modo da avere il numero (medio) ottimale di caratteri in ogni riga in relazione al font usato e al suo corpo 'normale'. Sotto molti aspetti questo pacchetto agisce come **typearea**, ma è decisamente più agile e consente di personalizzare in modo più esteso.

Ovviamente questi pacchetti sono utili anche nel creare nuove classi, perché basta caricarli all'inizio della nuova classe, specificando loro i parametri desiderati, così che spesso la creazione di una nuova classe si riduce veramente a poco.

13.4 Alcune classi non standard

Il sistema TeX viene distribuito con una enorme quantità di file aggiuntivi, compresi i file di classe che alcuni autori hanno predisposto e messo a disposizione degli utenti.

13.4.1 Le classi Komascript

Markus Kohm (da cui il prefisso 'Koma') ha messo a disposizione le quattro classi fondamentali per la composizione di libri, rapporti, articoli e lettere, rispettivamente *scrbook*, *scrreprt*, *scrartcl* e *scrlettr*. Le caratteristiche e i comandi disponibili sono reperibili nella documentazione del pacchetto **koma-script**.

Ma la componente di questo pacchetto certamente più utile è il pacchetto **typearea**, richiamabile esplicitamente dai file di classe, che permette di disegnarsi a proprio piacimento la griglia di scrittura secondo i criteri più diffusi; il pacchetto **typearea** è richiamabile anche da altri file di classe, non è una esclusiva dei file di classe di Markus Kohm.

13.4.2 La classe `memoir`

Peter Wilson ha scritto questa unica classe con la quale è possibile comporre quasi tutto quello che si può comporre con le classi standard, ma offre possibilità di personalizzazione difficili da realizzare se ci si appoggia alle classi standard.

Una parte pregevole di questo lavoro è costituita dal file contenente il manuale di documentazione, assai lungo e dettagliato, contenente anche una buona dose di consigli stilistici e una discreta storia del disegno grafico del libro e del documento; la lettura è decisamente interessante anche in queste parti meno tecniche; per le parti tecniche il manuale è essenziale perché l'estensione di LaTeX e delle sue classi che la classe *memoir* offre è decisamente importante.

La classe permette di personalizzare qualunque aspetto del documento, dal disegno grafico della pagina, alla scelta dei font per scrivere qualunque testo speciale, compresi i titolini delle sezioni, le testatine e i piedini. Qualunque cosa possa essere personalizzata, e che è difficile personalizzare usando LaTeX standard, con *memoir* può essere fatta facilmente e con poche istruzioni.

L'utente che si rivolge a *memoir* finisce con restarci affezionato per sempre; ma deve fare molta attenzione con le personalizzazioni, perché queste sono utili, certamente, ma come si può migliorare qualunque cosa, la si può anche peggiorare. In altre parole questa classe è decisamente più adatta ad un utente esperto, cosa che riesce a chiunque abbia un minimo di sensibilità estetica ed abbia spirito di osservazione per imparare e valutare i pregi e i difetti di ciò che viene fatto da altri. Dagli altri si dovrebbe prima di tutto imparare ad evitare gli errori, cercando di imitare il meglio e di avvicinarsi il più possibile alle realizzazioni degli artisti grafici più quotati; con lo strumento *memoir* ci si può riuscire.

L'autore di *memoir* nel suo manuale d'uso racconta anche cose che hanno a che vedere con la cultura tipografica e con il book design. Vale la pena di citare questo brano tradotto in italiano da Luciano Battaia.

> L'essenza di un libro ben stampato è che non si fa notare al primo, o addirittura al secondo o successivo, sguardo di chiunque non abbia un occhio allenato. Se la vostra prima reazione nello sfogliare un libro è di fare un'esclamazione di meraviglia osservando il layout, allora il libro è molto probabilmente mal progettato, se mai è stato progettato. La stampa di qualità è raffinata, non stridente.
>
> Con l'avvento del desktop publishing molti autori hanno la tentazione di progettare da soli i loro testi. Sembra molto facile farlo.

> Basta scegliere alcune delle migliaia di font disponibili, usarne uno per i titoli, uno per il testo principale, un altro per le didascalie, decidere le dimensioni dei caratteri, e la cosa è fatta.
>
> Tuttavia, come scrivere è un'abilità che bisogna apprendere, anche comporre tipograficamente un testo è un'arte che si deve apprendere e su cui bisogna esercitarsi. Ci sono centinaia di anni di esperienza racchiusi nel buon design di un libro. Essi non possono essere trascurati con leggerezza e molti autori che progettano i loro libri non conoscono alcune delle conquiste più importanti, per non parlare del fatto che quello che fanno è esattamente in antitesi con esse. Un esperto può infrangere le regole, ma allora sa che ha delle buone ragioni per farlo.
>
> [...] Se un libro grida 'guardami', questo è un avviso, e un pessimo avviso, per chi l'ha progettato.

13.4.3 Le tesi di laurea e la classe `toptesi`

Spesso gli studenti si avvicinano a LaTeX quando devono scrivere la tesi di laurea. Dagli archivi CTAN si possono scaricare diversi pacchetti che contengono il necessario per comporre la tesi di laurea o di dottorato. Fra i tanti si cita **toptesi** che contiene la classe *toptesi* oltre a varie utility e i loghi di numerose università italiane.

La classe consente di comporre tesi in italiano e in inglese: per comporre il frontespizio in lingua diversa dall'italiano si possono usare comandi specifici che possono essere inseriti in un file di configurazione; la tesi è personalizzabile per ogni lingua e per molti stili universitari. La classe è stata pensata anche per scrivere le tesi completamente in lingua diversa dall'italiano in vista del fatto che gli studenti in doppia laurea con i programmi Socrates/Erasmus devono scrivere la tesi anche (o solo) nella lingua dell'università ospitante.

Questa classe è una sovrastruttura della classe *report*, che, come si è detto, non distingue fra le parti inziali, centrali e finali del documento; la classe *toptesi* invece lo fa automaticamente: essa comincia a comporre con lo stile della `\frontmatter` della classe *book*, ma nel momento in cui viene eseguito il (nuovo) comando `\indici`, vengono composti i vari indici richiesti e poi viene eseguito l'equivalente del comando `\mainmatter`; con il comando `\appendix`, di fatto si passa alla composizione del materiale finale. Si noti che nella classe *book*, si prevede che il comando `\backmatter` venga eseguito *dopo* le appendici, non *prima*. Quel comando infatti eliminerebbe la dicitura 'Appendice ⟨*lettera*⟩' prima del titolo di ogni appendice e tra l'altro cessa di numerare le strutture a livello di 'capitolo'. La classe *toptesi* invece prevede che le appendici facciano parte del materiale finale, ma le distingue con la numerazione letterale. Ovviamente la questione se le appendici facciano o non facciano parte del materiale finale è una questione di gusti e di tradizione tipografica; il grande e citatissimo 'book designer' Bringhurst attribuisce le appendici al materiale finale. Chi ha predisposto la classe *book* attribuisce, invece, le appendici al materiale centrale del documento.

13.4.4 L'estensione `layaureo`

Fabiano Busdraghi ha prodotto un pacchetto di estensione che non modifica gli altri parametri della classe in uso, ma imposta il disegno grafico della pagina secondo il criterio della sezione aurea; il pacchetto si chiama **layaureo** e nel modo più semplice possibile, senza ricorrere ad impostazioni di parametri o di parole chiave talvolta 'immaginose', modifica la larghezza del testo e la sua altezza, nonché i margini di pagina in modo da ottenere il disegno grafico basato sulla sezione aurea.

In particolare allarga la giustezza in modo da coprire meglio la pagina A4 di quanto non lo facciano le classi standard; questo allargamento dipende dal corpo normale del font in uso. Poi se il documento deve essere composto sul recto e sul verso delle pagine, allora sistema il blocco del testo in modo che i margini stiano fra loro come la sezione aurea. È possibile usare una opzione per spostare tutto il testo di una quantità fissa verso l'esterno, per esempio 5 mm, al fine di tenere conto della rilegatura.

Si ricorda che la sezione aurea è un rapporto usato molto spesso nelle arti grafiche e lo si ritrova in moltissime strutture naturali; si veda in particolare l'equazione etichettata (sezione aurea) nella pagina 86.

Senza tenere conto dell'aumento eventuale del margine interno per tenere conto della rilegatura, la scelta della sezione aurea implica, per esempio, che il margine interno sia la frazione 0,618... del margine esterno.

13.5 I pacchetti di estensione

Anche nel paragrafo precedente si è parlato di alcuni pacchetti di estensione; virtualmente in ogni capitolo si è parlato di pacchetti di estensione.

I pacchetti di estensione, spesso chiamati *file di stile* a causa dell'estensione `.sty` che essi hanno, estendono le capacità di LaTeX mettendo a disposizione del compositore nuovi comandi o nuovi stili di composizione.

13.5.1 Come invocare i file di estensione

Nel preambolo di questo testo sono stati invocati diversi pacchetti di estensione; di alcuni si è già detto nel capitolo 4. Mentre i pacchetti descritti in quel capitolo sono da usarsi virtualmente ogni volta che si scrive un documento scritto con una tastiera nazionale (probabilmente in Italia con una tastiera italiana), con un alfabeto latino che contiene lettere accentate e in una lingua diversa dall'inglese, ora si mostrano le invocazioni degli altri pacchetti richiesti per la composizione di questo specifico testo:

```
\usepackage{pict2e}[2004/08/01]
\usepackage{graphicx}
\usepackage{mflogo}
\usepackage{amsmath,amsfonts,amssymb}
\usepackage{afterpage}
```

Il comando di invocazione è `\usepackage` che segue la sintassi seguente:

`\usepackage[`⟨*opzioni*⟩`]{`⟨*pacchetto*⟩`}[`⟨*data*⟩`]`

Le ⟨*opzioni*⟩ sono quelle specifiche del pacchetto invocato; per esempio per `babel` il pacchetto era stato invocato nel preambolo con il comando

```
\usepackage[italian]{babel}
```

Il ⟨*pacchetto*⟩ può consistere nel nome di un solo file senza la specificazione dell'estensione `.sty`, ma può anche consistere in una lista di nomi separati l'uno dall'altro da una virgola. La ⟨*data*⟩ è la data scritta secondo le norme ISO nella forma *aaaa/mm/gg* (anno con tutte quattro le cifre, mese indicato con due cifre anche se la prima è uno zero, giorno indicato con due cifre anche se la prima è uno zero) che indica la data rispetto alla quale quella del pacchetto deve essere uguale o posteriore; questa indicazione serve per essere sicuri di usare pacchetti sufficientemente aggiornati; come si vede nell'esempio riportato sopra, si è richiesto il pacchetto **pict2e** di data successiva o uguale al 1° agosto 2004, perché le versioni precedenti a questa data non disponevano di certi nuovi comandi introdotti in un secondo tempo.

Si fa notare che le ⟨*opzioni*⟩ indicate nell'invocazione del pacchetto sono *locali* a quel pacchetto e se questo non le riconosce viene emesso un avvertimento che informa che l'opzione tale-e-tale non è stata riconosciuta. Invece le ⟨*opzioni*⟩ specificate nella invocazione del file di classe attraverso il comando `\documentclass` sono *globali*, nel senso che esse vengono passate anche alle successive invocazioni dei pacchetti; l'opzione *italian* specificata per **babel** avrebbe potuto essere indicata anche fra le opzioni di `\documentclass` e così avrebbe potuto essere usata, senza bisogno di ripeterla, anche da altri pacchetti, come per esempio **varioref** o **layout**; il primo esegue le citazioni nella forma, per esempio, '... nella figura "3.5 della pagina 123"...'; il secondo pacchetto serve per disegnare in una pagina a se stante il layout grafico della pagina, indicando graficamente i rettangoli che descrivono la gabbia, o la testatina, e a parole le indicazioni metriche che nel disegno sono riportate solo mediante delle frecce.

13.5.2 I vari pacchetti e gli archivi internazionali

È facile comprendere che i pacchetti disponibili siano numerosissimi; l'archivio internazionale CTAN (che sta per Comprehensive TEX Archive Network) ne contiene diverse migliaia ottenuti come contributi dei vari utenti di LATEX che, come succede con il software libero, hanno realizzato qualche estensione per risolvere qualche loro problema contingente e hanno ritenuto che fosse utile anche per per gli altri utenti; perciò hanno caricato il pacchetto nell'archivio in modo che chiunque potesse servirsene liberamente e gratuitamente.

Il sistema degli archivi CTAN è formato da tre archivi principali, uno negli USA, uno nel Regno Unito e uno in Germania, che sono sempre sincronizzati l'uno con l'altro e costituiscono la sorgente principale delle estensioni e delle distribuzioni gratuite del sistema TEX. Sparsi per il mondo ci sono poi innumerevoli siti che costituiscono i *mirror* dei tre archivi principali; questi mirror vengono sincronizzati dai loro amministratori quasi in tempo reale, ma generalmente non sono in ritardo di più di una settimana rispetto ai tre archivi principali. Per questo motivo c'è sempre la possibilità di avere un archivio abbastanza vicino così che le connessioni internet siano le più rapide possibile.

Tutti gli archivi e quasi tutti i mirror hanno una pagina interattiva che permette di cercare i file che interessano o per nome o per argomento; solitamente

sono anche predisposti per permettere di scaricare intere cartelle in formato compresso così da essere sicuri di non dimenticare nulla per la strada.

Le distribuzioni migliori, come MiKTeX per le macchine Windows, dispongono di un wizard, altri di un assistant, altri di un package manager, che permettono di avere preliminarmente il nome di tutti i pacchetti disponibili e di scaricare sul proprio calcolatore solo i pacchetti che interessano.

Gli archivi principali sono:

`cam.ctan.org`
`dante.ctan.org`
`tug.ctan.org`

In Italia l'unico mirror registrato è quello dell'Università di Roma Tor Vergata all'indirizzo

`cis.uniRoma2.it`

ma, specialmente dall'Italia settentrionale, lingua permettendo, si può accedere ai mirror francesi, svizzeri, austriaci e, in particolare all'archivio dell'associazione Dante, localizzato nella Germania meridionale. Nel seguito con CTAN si indicherà indifferentemente l'insieme degli archivi internazionali o semplicemente l'indirizzo di base di uno degli archivi principali o di un mirror.

Cercando bene si può quasi sempre trovare il problema già risolto in un qualche pacchetto presente negli archivi.

13.6 Come scrivere nuovi pacchetti

Tuttavia talvolta può essere necessario crearsi un pacchetto di estensione personale; per questo scopo bisogna leggere con attenzione la già citata guida `clsguide` nella cartella del proprio sistema dedicata alla documentazione `.../doc/latex/`; si scoprirà che la cosa di per sé è facilissima; se c'è qualche difficoltà questa risiede nella scelta di che cosa mettere nel pacchetto personale.

C'è però una risposta semplicissima; tutti hanno bisogno di estensioni personali e prima o poi si scrivono dei nuovi comandi ad uso personale, che rispecchiano il proprio stile di gestione del file sorgente. Possono essere macro per inserire certi loghi o certi marchi che hanno a che fare con l'istituzione alla quale si appartiene; possono essere macro per scrivere sigle o abbreviazioni frequenti nei testi che si scrivono abitualmente; possono essere nuovi ambienti che riflettono le proprie necessità compositive; possono essere anche semplici modifiche di comandi standard.

Bene, tutto ciò può essere collocato in un file chiamato in una maniera un po' personalizzata, diversamente dal solito `myfile.sty`, per esempio `estensionedi`⟨*iniziali*⟩`.sty`, dove ⟨*iniziali*⟩ può consistere nelle iniziali del proprio nome e cognome o quant'altro personalizzi il vostro file. L'estensione del nome del file *deve* essere `.sty`.

Seguendo le istruzioni della Guida, il file deve cominciare con due righe importantissime:

```
\NeedsTeXFormat{⟨formato⟩}[⟨data⟩]
\ProvidesPackage{⟨nome-del-pacchetto⟩}[⟨versione e descrizione⟩]
```

Il ⟨*formato*⟩ se usate LaTeX 2ε, come si suppone, sarà `LaTeX2e`; la ⟨*data*⟩ sarà la data del formato che voi avete usato per scrivere le vostre macro; l'indicazione di questa data serve per assicuravi che il vostro file per sbaglio non possa venire usato da voi (su un'altra macchina) o da altri con una versione obsoleta del ⟨*formato*⟩; la data, scritta nella forma ISO, cioè *aaaa/mm/gg*, può essere indicata opzionalmente anche nella riga nella quale si richiama con `\usepackage` un pacchetto qualsiasi, in modo da essere sicuri di non usare versioni obsolete; naturalmente dovete conoscere la data del pacchetto non obsoleto che vi serve.

Il ⟨*nome-del-pacchetto*⟩ sarà verosimilmente il nome del vostro file personale, mentre la ⟨*versione e descrizione*⟩ deve essere una stringa di parole, possibilmente di lunghezza non maggiore di una riga di caratteri della finestra comandi, o del terminal, o della console; queste informazioni devono avere la forma seguente:

⟨*aaaa/mm/gg*⟩ `v.`⟨*versione*⟩ ⟨*breve descrizione*⟩

La data è quella della creazione del pacchetto o della sua ultima revisione in formato ISO. La versione potete scriverla nella maniera che preferite. La breve descrizione deve essere una breve frase del tipo "Macro di Mario Rossi per comporre il manuale di tricotetratomia".

Dopo queste due righe potete inserire tutte le definizioni che volete e/o richiamare i pacchetti che volete; terminate il file con `\endinput`.

L'argomento delle definizioni verrà trattato nel capitolo 16.

13.7 Non modificare i pacchetti esistenti

Alcuni utenti di LaTeX credono di poter risolvere i loro problemi compositivi andando a modificare i file di classe o i pacchetti presenti nella propria distribuzione del sistema TeX o scaricati da CTAN.

Questo deve essere assolutamente evitato! Non lo si ripeterà mai abbastanza, ma questa pratica, oltre ad essere vietata dalla licenza a cui sono sottoposti quasi tutti i pacchetti esistenti, è una pratica autolesionista.

I modi corretti di procedere sono i seguenti.

1. Se le modifiche da apportare alla classe o al file di stile sono numerose, il modo migliore di procedere è quello di copiare il file in una cartella dell'albero personale `.../tex/latex/`⟨*cartella*⟩. Il nome di questa ⟨*cartella*⟩ verrà scelto in modo mnemonico, ma diverso dai nomi delle cartelle già esistenti; il file copiato verrà ribattezzato con un altro nome; per esempio se si vuole modificare la classe *`book`* si copierà il file `book.cls` nella propria ⟨*cartella*⟩ dandole, per esempio, il nome `tttomia.cls`.

   In questo nuovo file cambiate il contenuto di `\ProvidesClass`, per esempio modificate la dichiarazione

   ```
   \ProvidesClass{book}
                 [2004/02/16 v1.4f
    Standard LaTeX document class]
   ```

   nel modo seguente

```
\ProvidesClass{tttomia}
              [2007/03/10 v1.0
 Mario Rossi: classe per il libro di TricoTetraTomia]
```

In questo nuovo file apportate tutte le modifiche che desiderate e aggiungeteci tutti i comandi che volete usare; questo siete liberi di farlo secondo la licenza delle classi, ma non siete liberi di modificare il file originale senza cambiargli il nome.

Nello stesso tempo voi potete usare la classe standard in qualunque altra circostanza; questa inoltre vi potrà servire per seguire la stessa strada, con nomi diversi, se volete adattare la classe *book* alla composizione di un altro libro.

Lo stesso procedimento può essere seguito per modificare i file di stile.

2. Potete modificare solo qualche comando che vi interessa in modo particolare; copiate dal file di classe originale o dal file di stile i comandi che volete modificare e incollateli nel vostro file `mymacros.sty` di comandi e macro personali; sostituite il comando `\newcommand` con `\renewcommand` e fate operazioni simili per le definizioni degli ambienti. Nel corpo della definizione inserite tutte le modifiche che volete.

Ricordate che i comandi definiti dentro il file di classe o di stile originali contengono spessissimo la 'lettera' @; è lecito usare questo segno come se fosse una lettera dell'alfabeto solo all'interno dei file di classe o di stile; quindi anche all'interno del vostro file `mymacros.sty`. State attenti però a non combinare pasticci; modificare o ridefinire comandi che contengano @ nel loro nome vi espone al rischio di modificare un comando di sistema in modo errato e tutto il vostro sistema TEX potrebbe diventare inutilizzabile; la riparazione è semplicissima, perché vi basta tornare sui vostri passi eliminando dal vostro file le modifiche o le definizioni che potrebbero avere sconvolto il sistema, e poi potete ricominciare con più attenzione stando bene attenti a che cosa ridefinite o modificate. Usate sempre i comandi `\newcommand` e `\renewcommand` che controllano la precedente esistenza di comandi con lo stesso nome; potete usare per queste verifiche anche i comandi di sistema `\@ifdefinable` e `\@ifundefined`; le rispettive sintassi sono le seguenti:

`\@ifdefinable{`⟨*comando*⟩`}{`⟨*azione*⟩`}`
`\@ifundefined{`⟨*nome del comando*⟩`}{`⟨*azione*⟩`}`

Il comando `\@ifdefinable` verifica che il ⟨*comando*⟩ che si vorrebbe definire sia definibile; cioè che (*a*) non sia stato già definito, (*b*) sia diverso da `\relax`, (*c*) non cominci con `\end`. Se sono verificate queste condizioni, allora si può eseguire l'⟨*azione*⟩, verosimilmente la definizione vera e propria del ⟨*comando*⟩.

Invece `\@ifundefined` verifica che il ⟨*nome del comando*⟩ (il comando privato dell'iniziale backslash) sia davvero una entità priva di definizione, e se è verificata questa condizione esegue l'⟨*azione*⟩, verosimilmente la definizione.

Evitate di usare i comandi primitivi di definizione come `\def`, `\edef`, `\gdef` e `\xdef`, di cui si parlerà più avanti nel capitolo 16, perché questi comandi eseguono le definizioni in modo incondizionato e rischiate di azzoppare il sistema. Per altro voi vedrete che nei file di classe e di stile questi comandi sono usati spessissimo; lasciateli dove sono ma non usateli voi stessi.

Capitolo 14

BibTeX: la bibliografia

14.1 Introduzione

Come si è già detto, LaTeX compone la bibliografia come un testo speciale costituito da un elenco di voci etichettate con un'etichetta di vari stili, ma citabili con una chiave mnemonica scelta dall'autore o fissata in altro modo.

Il problema della bibliografia non è tanto quello di preparare l'elenco con la sintassi descritta per l'ambiente *thebibliography*, quanto quello di elencare i riferimenti in un ordine logico, nello scrivere tutte le informazioni necessarie al reperimento dell'opera citata, e, specialmente, nell'essere consistenti nello stile di presentazione di tutte queste informazioni.

14.2 Il programma BibTeX

Il programma BibTeX serve per rispondere alle necessità descritte sopra in modo tipograficamente perfetto e versatile.

Questo programma richiede che si disponga di un database di riferimenti bibliografici scritto in una certa maniera che fra poco verrà sommariamente illustrata; esso trae da questo database solo i riferimenti citati in un dato documento e ne prepara l'elenco, generalmente ordinato alfabeticamente secondo il cognome del primo autore, nella forma corretta per essere trattato da LaTeX e composto secondo quanto desiderato.

14.2.1 Come specificare lo stile bibliografico

Nel preambolo del documento bisogna specificare lo stile compositivo dell'elenco bibliografico; lo si fa con il comando `\bibliographystyle` secondo la seguente grammatica:

`\bibliographystyle{`⟨*stile bibliografico*⟩`}`

Eventualmente questo comando può essere preceduto dalla specificazione di un pacchetto che interagisce con la composizione della bibliografia mettendo a disposizione diverse varianti dei comandi di citazione, per esempio il pacchetto **natbib**.

14.2.2 Come comporre la bibliografia

Si inserisce il comando `\bibliography` con la seguente sintassi nel punto dove si vuole comporre la bibliografia

`\bibliography{`⟨*database bibliografico*⟩`}`

dove ⟨*database bibliografico*⟩ è il nome del database bibliografico che si intende usare; disponendo di più database e dovendo citare opere elencate in diversi database, il ⟨*database bibliografico*⟩ può consistere in una lista di nomi separati da virgole.

Gli stili bibliografici distribuiti di default sono file con l'estensione `.bst` e si trovano nella cartella `.../bibtex/bst/` dove sono raccolti in diverse sottocartelle. Gli stili più diffusi sono `unsrt` e `alpha`; il primo non esegue nessun ordinamento, ma le opere sono elencate nello stesso ordine in cui vengono citate; con il secondo stile le opere vengono ordinate secondo l'ordine alfabetico del primo o unico autore, o quanto possa sembrare il nome di un autore al programma, poi per anno e poi per ordine alfabetico del titolo. Con questi stili le opere vengono citate per numero d'ordine racchiuso fra parentesi quadre, come si fa abitualmente nei lavori tecnico–scientifici. Per citazioni diverse bisogna affidarsi al pacchetto di estensione **natbib** alla cui documentazione si rinvia per i dettagli necessari.

Tutti questi stili ipotizzano che la bibliografia sia scritta in inglese; non è difficile crearsi degli stili personalizzati, in particolare in italiano, ricorrendo al file `makebst.tex` che provvede a tutto il necessario pur di rispondere 'correttamente' ad un certo numero di domande; questo file va elaborato con LaTeX e richiede che si modifichi un file modello dal nome `merlin.mbs`; tutta l'operazione non è difficile, ma richiede che uno abbia già acquisito una certa dimestichezza con i database bibliografici e con i vari stili bibliografici.

14.2.3 Chiavi e citazioni

Ma la cosa non finisce qui; per avere la bibliografia composta il programma deve disporre delle chiavi di citazione; perciò bisogna lanciare LaTeX; appena ha finito e non ci sono errori, tranne eventualmente avvisi che alcune chiavi di citazione non sono state 'risolte' (il che significa che LaTeX non ha trovato altro che le chiavi, cioè gli argomenti dei comandi `\cite`, ma non sa a che cosa corrispondano queste chiavi) allora bisogna passare alla fase successiva.

A questo punto bisogna lanciare BibTeX o da una finestra comandi o terminal o console, oppure bisogna cliccare sull'apposito bottone dello shell editor. In questo modo BibTeX apre il o i database specificati, cerca le voci corrispondenti alle opere citate in base alle chiavi raccolte da LaTeX nel giro precedente; le ordina secondo i criteri di ordinamento specificati nello stile bibliografico e finalmente genera un file con estensione `.bbl` che contiene in linguaggio LaTeX tutto quello che occorre per comporre la bibliografia.

Finalmente si lancia nuovamente LaTeX e questa volta viene composta la bibliografia e vengono decifrate le chiavi; ma non è finita; le chiavi sono decifrate, ma non sono ancora state usate per le citazioni; bisogna lanciare ancora una volta LaTeX in modo che possa usare le chiavi decifrate al giro precedente così che le informazioni ci siano tutte e siano state usate tutte al posto giusto. Una ulteriore passata del documento a LaTeX non guasta; serve per giustificare

correttamente il documento e può rendere corretti i riferimenti incrociati che avrebbero potuto risultare modificati dall'uso delle chiavi.

Esercizio 14.1 Perché dopo la seconda volta che si lancia LaTeX dopo aver lanciato BibTeX i riferimenti incrociati potrebbero non essere corretti? Perché quindi è necessario lanciare LaTeX ancora una volta?

In sostanza la generazione e composizione della bibliografia avviene in quattro o cinque fasi così:

(1) LaTeX (2) BibTeX (3) LaTeX (4) LaTeX (5) LaTeX

Non è il caso di spaventarsi di questa (apparente) complicazione; bisogna ricordare prima di tutto che un buono shell editor dispone quasi sempre di un bottone da cliccare che esegue tutte le passate che occorrono per avere la bibliografia composta correttamente e i riferimenti incrociati fra loro perfettamente coerenti; a cosa servirebbe sennò un buono shell editor?

Nello stesso tempo, se lo shell editor non disponesse di quel magico bottone e fosse necessario procedere a mano attraverso la finestra comandi o il terminal o la console, eseguire quei cinque comandi richiede una manciata di secondi; la compilazione di un testo di molte centinaia di pagine, diciamo 700, richiede con i PC o laptop moderni qualche decina di secondi, BibTeX richiede qualche secondo; l'intera operazione da eseguirsi una volta sola alla fine della lavorazione del documento può richiedere un paio di minuti; siccome anche l'indice analitico richiede lo stesso numero di passate, si può ottimizzare l'operazione prendendo due piccioni con una fava. Non è terribile.

14.3 I database bibliografici

I database bibliografici sono dei file scritti in testo puro con i caratteri ASCII o anche con i caratteri accentati gestibili con l'input encoding preferito; per la trasportabilità sarebbe meglio che si trattasse di caratteri ASCII puri (codici a 7 bit) e quindi i segni letterali con diacritici dovrebbero essere rappresentati con le sequenze per gli accenti che non sono state descritte finora da nessuna parte, ma si rinvia a qualunque guida o manuale 'oldstyle' di LaTeX; qui non si vuole fare gli schizzinosi circa le tradizionali modalità di scrivere le lettere accentate immaginando di usare una tastiera USA, priva di tasti speciali per i caratteri accentati; semplicemente si ritiene molto più comodo servirsi delle potenzialità del proprio PC o laptop per gestire i caratteri con diacritici e le potenzialità dei file di estensione di LaTeX per comporre correttamente quelle lettere speciali.

Il punto è che in Europa, e in Italia in particolare, è difficile comprare un PC o un laptop con una tastiera diversa da quella nazionale. In Italia, in particolare, scrivere le vecchie sequenze LaTeX per gli accenti sarebbe particolarmente fastidioso perché l'accento italiano più diffuso è quello grave e dalla tastiera italiana manca qualunque tasto o combinazione di tasti che consenta di introdurlo; bisogna ricorrere al tastierino numerico, il che con i laptop si può fare ma richiede ulteriori tasti da premere.

Dopo questa lunga premessa del fatto che sarebbe meglio usare i caratteri ASCII, ma che con la tastiera italiana la cosa è particolarmente fastidiosa, veniamo al database; ogni voce del database comincia con la dichiarazione del

tipo di documento a cui la voce si riferisce; questa dichiarazione è una delle seguenti parole: `article`, `book`, `booklet`, `conference`, `inbook`, `incollection`, `inproceedings`, `manual`, `masterthesis`, `misc`, `phdthesis`, `proceedings`, `techreport`, `unpublished`. Ognuna di queste categorie di documenti è spiegata nella documentazione di BIBTEX, in `.../doc/bibtex/btxdoc.dvi`.

In base alle norme ISO ognuna di queste tipologie di documento richiede certe informazioni, quindi nel descrivere il documento bisogna usare un certo numero di parole chiave; se per un certo tipo di riferimento una voce obbligatoria manca, quando BIBTEX viene eseguito, vengono emessi degli avvertimenti del fatto che la citazione tale è priva della specificazione talaltra. Le chiavi facoltative e obbligatorie per ogni tipologia sono specificate nella documentazione citata.

Qui ci limiteremo a qualche piccolo esempio commentando ciò che appare nelle voci del database.

```
@manual{b:refbib,
title=       "Documentation - {B}ibliographical references -
             {E}ssential and supplementary  elements",
note=        "Norma {ISO} 690--1975",
organization="{International Organization for Standardization}",
address=     "Ginevra",
year=        "1975",
key=         "{ISO 690}"
}

@manual{b:patashnik,
author=      "Oren Patashnik",
title=       "\textsc{Bib}{\TeX}ing",
organization="{TUG}",
year=        "1988",
note=        "Il file \texttt{btxdoc.dvi} contenente questa
           documentazione generalmente è già presente nella
           cartella \texttt{/texmf\slash doc\slash bibtex/}
           poiché fa parte integrante del sistema \TeX"
}

@book{b:latexe,
author=      "Leslie Lamport",
title=       "{\LaTeX, a document preparation system ---
             User's guide and reference manual}",
publisher=   "{Addison--Wesley Publ. Co.}",
address=     "{Reading, Mass.}",
year=        "1994",
edition=     "2"
}
```

Come si vede dai tre esempi di voci bibliografiche riportati sopra, la tipologia del documento viene sempre preceduta dal segno `@` e il contenuto della voce è racchiuso fra parentesi graffe. Il primo elemento contenuto fra queste parentesi è la chiave di citazione; nel file LATEX, quindi, le norme ISO per le citazioni bibliografiche *devono* essere citate con `\cite{b:refbib}` e non con un'altra chiave che piaccia di più.

Le altre informazioni relative al documento citato sono specificate mediante una espressione del tipo

⟨*parola-chiave*⟩ `="`⟨*testo*⟩`",`

dove l'ultima virgola può essere omessa solo nell'espressione finale. Ciò che nel ⟨*testo*⟩ compare fra graffe non subisce modificazioni quando viene trascritto nel file di uscita; altrimenti BibTeX spesso cambia le maiuscole in minuscole con il risultato di alterare quello che si vuole scrivere, oppure di cambiare l'ortografia di macro che verranno poi interpretate da LaTeX.

Le norme ISO sono classificate come `manual`; questa tipologia non ha bisogno dell'indicazione dell'autore; per consentire un qualunque ordinamento, allora, è opportuno inserire una parola chiave `key` che serve come chiave di ordinamento in mancanza di altre chiavi predefinite per questo scopo.

Anche la documentazione di BibTeX è classificata come manuale, ma l'autore è noto ed indicato, per cui non è necessaria una chiave supplementare.

L'ordine in cui vengono scritte le ⟨*parole-chiave*⟩ e le relative espressioni non è essenziale; l'importate è che ci siano tutte quelle relative ad una data tipologia ed eventualmente ci siano quelle facoltative che possono agevolare l'ordinamento secondo qualche criterio dipendente dallo stile bibliografico prescelto.

Dipende dal compositore decidere se conviene predisporsi e tenere aggiornato uno o più database bibliografici oppure se comporre di volta in volta la lista dei riferimenti mediante la compilazione del contenuto dell'ambiente *thebibliography*; inizialmente si è un po' restii a creare un database bibliografico; poi se si supera questo scoglio iniziale se ne apprezza completamente la validità e la comodità.

Capitolo 15

LaTeX: indici e glossari

15.1 Introduzione

Gli indici analitici e i glossari possono essere molto utili nelle opere di consultazione. LaTeX offre un grande aiuto per la compilazione di questi elenchi, ma la parte difficile resta quella dell'autore che deve decidere che cosa inviare all'indice analitico o al glossario, se deve usare voci indipendenti o subordinate, se deve distinguere il modo di scrivere le pagine a seconda di come il lemma di quella voce viene trattato; definizione, esempio d'uso, applicazione, eccetera.

15.2 L'indice analitico

L'indice analitico è un elenco di voci ordinate alfabeticamente anche con alcuni livelli di subordinazione, e vicino ad ogni lemma viene inserito il numero di pagina dove quel lemma viene trattato. Il lemma può essere trattato in diversi punti del testo, quindi di fianco al lemma può comparire un elenco di numeri di pagina.

La raccolta di queste informazioni viene fatta mediante il comando `\index`; la sintassi è

```
\index{⟨lemma⟩}
```

Se nel prologo c'è l'istruzione `\makeindex`, l'informazione specifica relativa al ⟨*lemma*⟩ viene inviata ad un file ausiliario da elaborare in un secondo tempo; se quell'istruzione manca nel file ausiliario non viene scritto niente. Quando i calcolatori erano lenti, era importante non rallentare l'elaborazione del file sorgente con operazioni non strettamente necessarie; oggi la cosa non è più così importante, ma visto che l'indice analitico è l'ultima cosa che si compone in un libro, è bene non perdere tempo con questa informazione finché non si è alla fine o quasi alla fine.

L'informazione che viene scritta da LaTeX nel file ausiliario (il cui nome ha l'estensione `.idx`) è del tipo:

```
\indexentry{⟨lemma⟩}{⟨pagina⟩}
```

Per ottenere questa raccolta di dati basta collocare l'istruzione `\index` con il suo argomento 'attaccata', cioè senza lasciare spazi, alla parola che si vuole indicizzare. Per esempio:

```
... la tecnologia dei transistori\index{transistore}
      NPN\index{transistore!NPN} ...
```

15.2.1 Il programma makeindex

Per elaborare il file `.idx` in modo da metterlo nella forma giusta, cioè in ordine alfabetico, strutturato per lemmi, sotto-lemmi e sotto-sotto-lemmi, con gli elenchi di numeri di pagine senza doppioni o con intervalli, eventualmente composti con caratteri diversi a seconda dell'uso del lemma, esiste il programma makeindex; si tratta di un programma da lanciare a mano dalla finestra comandi o dal terminal o dalla console oppure premendo l'opportuno pulsante dello shell editor.

Questo programma esamina il ⟨*lemma*⟩ come trascritto senza modifiche dall'argomento di `\index` al file `.idx` e lo elabora secondo quanto vi trova indicato.

Infatti quello che appare in ⟨*lemma*⟩ non è necessariamente il solo 'lemma' ma anche informazioni ausiliarie che facilitano il compito del programma makeindex.

Nel primo esempio elementare indicato sopra, il ⟨*lemma*⟩ è costituito semplicemente dalla parola 'transistore'; questa parola serve a diversi scopi:

1. indica che si tratta di un lemma di primo livello;
2. indica che la chiave con cui eseguire l'ordinamento alfabetico è 'transistore';
3. indica che va riportato nell'indice analitico tale e quale, vale a dire stampato con il font di default con cui viene composto l'indice;
4. che il numero della pagina in cui compare va scritto anch'esso con il font di default.

Strutturando il ⟨*lemma*⟩ in modo più articolato le varie funzioni indicate nella precedente enumerazione possono essere trattate in modo diverso così da far apparire il livello del 'lemma' nell'indice, per indicare una diversa chiave di ordinamento, per indicare come scrivere il 'lemma' nell'indice e come stampare il numero della pagina; la strutturazione consente anche di indicare l'inizio e la fine degli intervalli di pagine, per inserire nell'indice il rinvio ad un altro lemma, eccetera.

I dettagli sono spiegati bene nella documentazione del programma makeindex che si trova nella cartella `.../doc/makeindex/` dove non solo è contenuto il file `makeindex.dvi` ma anche un semplice tutorial `ind.dvi`.

Sommariamente qui si riportano due esempi:

1. Scrivendo

   ```
   \index{transistore!NPN}
   ```

   il lemma 'NPN' viene scritto strutturato sotto al lemma 'transistore'.

2. Scrivendo

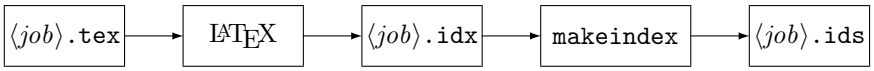

Figura 15.1: Le fasi per la produzione dell'indice analitico

```
\index{ambiente!enumerate@\texttt{enumerate}}
```

il lemma 'enumerate' viene scritto con il font a spaziatura fissa e strutturato sotto al lemma 'ambiente'.

Naturalmente per sfruttare tutta la forza del programma makeindex bisogna leggere attentamente la documentazione citata e prendere esempio dal tutorial menzionato sopra.

15.2.2 La composizione effettiva dell'indice analitico

La composizione effettiva dell'indice analitico avviene specificando nel file sorgente per LaTeX:

```
\input{⟨file⟩.ind}
```

dove ⟨*file*⟩ è il nome del 'job' e `.ind` ne è l'estensione; Il 'job' coincide con il nome del file sorgente per LaTeX che ha dato luogo alla raccolta dei dati per l'esecuzione dell'indice analitico ed è stato elaborato da `makeindex` secondo lo schema della figura 15.1. Questo file contiene anche i comandi di apertura e di chiusura dell'ambiente *theindex*, quindi se non si vuole altro, basta semplicemente eseguire l'inclusione di questo file tramite il comando `\input`.

Come la figura dimostra chiaramente, per ottenere la composizione definitiva dell'indice analitico bisogna lanciare LaTeX ancora una volta, perché finché il file ⟨*job*⟩`.ids` non è disponibile, l'indice non può essere composto.

Usando il pacchetto di estensione **mkindex** è possibile servirsi del semplice comando `\printindex` che provvede da solo a generare l'ambiente e a comporre l'indice analitico.

Siccome i numeri delle pagine devono essere quelli definitivi, è evidente che la composizione dell'indice analitico deve essere in assoluto l'ultima operazione da fare; se si deve eseguire una qualche correzione nel testo, questa correzione comporta anche la correzione dell'indice analitico, vale a dire la triplice esecuzione di LaTeX con l'esecuzione di makeindex dopo la prima di LaTeX. Per fortuna gli shell editor hanno generalmente un bottone solo che esegue tutta questa procedura, per altro necessaria anche per comporre la bibliografia usando BibTeX.

15.3 Il glossario

Un glossario è un elenco di voci accompagnate da una breve spiegazione; talvolta appare anche la pagina del testo dove viene riportata una spiegazione meno succinta.

Il procedimento per produrre un glossario è simile a quello usato per comporre l'indice analitico, ma è un po' più artigianale, come si vedrà fra poco.

L'inizio consiste nel marcare il testo sorgente in una maniera simile a quella dell'indice analitico:

```
\glossary{⟨voce⟩}
```

Lanciando LaTeX, se il preambolo contiene l'istruzione `\makeglossary`, viene prodotto anche un file ⟨*job*⟩`.glo` che contiene tutte le voci di cui si vuol generare il glossario accompagnate dal numero della pagina dove esse compaiono insieme ad una loro spiegazione non succinta.

Lanciando il programma makeindex dalla linea di comando

```
makeindex -s gglo.ist -o ⟨job⟩.gls ⟨job⟩.glo
```

si produce l'elenco alfabetizzato ⟨*job*⟩`.gls` delle parole di cui si desidera comporre il glossario;

Bisogna aprire questo file e aggiungere la breve spiegazione oltre che a correggere la forma di stampa del numero di pagina (per esempio per cancellarlo, se esso non svolge nessuna funzione specifica, come quando il glossario non è altro che un elenco di simboli o di abbreviazioni); poi bisogna comporlo in una maniera simile a quello che si fa per la composizione dell'indice analitico.

In questo senso la composizione del glossario è un po' più artigianale, in quanto bisogna intervenire ancora sul file alfabetizzato.

Esistono diversi pacchetti in rete per comporre glossari o simili elenchi commentati. Qui si suggerisce il pacchetto **glossary** scaricabile dagli archivi CTAN; esso offre una maniera comoda per scrivere la ⟨*voce*⟩ in modo che contenga direttamente anche la descrizione della voce, così che quando si compone il file alfabetizzato esso sia già pronto per essere composto con LaTeX. Il pacchetto, ovviamente, definisce il comando `\printglossary`, del tutto simile nelle sue funzioni a `\printindex`.

Si raccomanda la lettura attenta della documentazione del pacchetto **glossary** perché esso offre diverse altre funzioni avanzate che possono tornare molto comode per la generazione di un glossario e per l'amministrazione delle voci da inserirvi.

15.4 Come modificare la composizione dell'indice analitico

Quanto viene esposto qui vale anche per modificare la composizione del glossario.

Per entrambi il file che compone l'elenco richiesto è prodotto dall'esecuzione del programma makeindex; per l'indice analitico viene prodotto il file ⟨*job*⟩`.ind` mentre per il glossario viene prodotto il file ⟨*job*⟩`.gls`. entrambi questi file racchiudono l'elenco da comporre dentro due specifici ambienti *theindex* e *theglossary*. Come ogni voce sia articolata dipende da come si siano usati i comandi `\index` o `\glossary`, e quindi da come le varie opzioni siano state elaborate da makeindex per produrre i file di uscita. Cambiando opzioni, chiaramente, cambia anche la composizione.

Tuttavia qui si vuole richiamare l'attenzione su un altro punto e per esemplificare ci riferiremo all'indice analitico; per il glossario valgono considerazioni del tutto analoghe.

La formattazione dell'indice analitico dipende da come è definito l'ambiente *theindex* oltre che dal contenuto del file ⟨*job*⟩`.ind`. Cambiare formattazione dell'indice analitico, implica la capacità di cambiare la definizione dell'ambiente

theindex. Si tratta di modificare la definizione di un ambiente di sistema e l'operazione, benché non difficile, richiede un minimo di attenzione e di conoscenza. L'argomento viene rimandato al paragrafo 16.8 nel capitolo 16.

Capitolo 16

LaTeX: nuovi comandi

16.1 Introduzione

Uno dei punti di forza del linguaggio di programmazione che sostiene il sistema TeX è costituito dal fatto di poter definire nuovi comandi, nuove istruzioni.

L'interprete TeX o pdfTeX che interpreta i comandi dei vari tipi di mark-up dei file sorgente è in grado di eseguire alcune centinaia di comandi base, comandi primitivi, non nel senso che siano rozzi, ma che sono quelli a cui tutti gli altri si riferiscono; i comandi derivati sono per lo più definiti nei vari file di formato; `plain.tex` per il mark-up gestito da plain TeX; `latex.ltx` per il mark-up di LaTeX, eccetera.

Questi file di formato *definiscono* dei nuovi comandi, detti *macro*, attraverso i comandi primitivi o quelli derivati già definiti.

Quando l'interprete incontra una macro la deve sviluppare, cioè deve ricercarne in memoria la traduzione in base alla definizione specifica di quella macro; se questa definizione contiene le azioni da eseguire specificate solamente mediante comandi primitivi, l'interprete li esegue, altrimenti sequenzialmente ricerca i comandi derivati presenti nella definizione, ripetendo il processo finché lo sviluppo contiene solo comandi primitivi che finalmente esegue.

Ogni macro può essere definita in modo che possa lavorare su *argomenti*, cioè parti variabili che cambiano di volta in volta che si invoca la macro. Gli argomenti possono essere *delimitati* o *non delimitati*. All'interno dei formati si fa un grande uso di argomenti delimitati, ma i comandi per gli utenti, specialmente con LaTeX, fanno un uso minimo di argomenti delimitati.

Gli argomenti non delimitati sono costituiti da un solo *token*, per esempio una sola lettera o una sola cifra, e non richiedono di venire racchiusi dentro nessuna struttura che ne limiti la lunghezza. Altrimenti devono essere racchiusi fra parentesi graffe. Nei capitoli precedenti si sono visti quasi sempre argomenti delimitati indicati nelle istruzioni sintattiche con `{`⟨*argomento*⟩`}`. LaTeX usa raramente gli argomenti delimitati, ma li delimita solo mediante parentesi quadre; è quanto si è indicato con `[`⟨*argomento*⟩`]`; nell'ambiente *picture* sono delimitate le coordinate, racchiuse fra parentesi tonde e indicate nelle costruzioni sintattiche con `(`⟨*arg-1*⟩`,`⟨*arg-2*⟩`)`. Ma per il resto è difficile trovare altri comandi destinati al compositore che facciano uso di argomenti delimitati.

Qui si illustreranno succintamente le definizioni di nuovi comandi, le ridefi-

nizioni di comandi già esistenti; le definizioni di nuovi ambienti e le ridefinizioni di ambienti già esistenti.

16.2 Le definizioni di comandi nuovi

Per definire un nuovo comando si usa la seguente sintassi:

```
\newcommand{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
\newcommand*{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
```

dove ⟨*num-argomenti*⟩ è il numero degli argomenti su cui opera la ⟨*macro*⟩ dei quali il primo può essere facoltativo; esso è facoltativo se viene espressa anche la seconda opzione incluse le sue parentesi quadre; se nell'invocazione della macro che accetta un argomento facoltativo non lo si specifica, esso riceve il valore di ⟨*default*⟩. Nella definizione della ⟨*macro*⟩ gli argomenti, al massimo in numero di 9, sono indicati con il loro numero progressivo `#1`, `#2`, eccetera. La ⟨*macro*⟩, cioè il suo nome, può essere costituita solo:

- dal segno di backslash `\` seguito da una stringa formata solamente da lettere dell'alfabeto latino maiuscole o minuscole, ricordando che LaTeX distingue le une dalle altre; oppure
- dal segno di backslash `\` seguito da un solo segno non alfabetico; oppure
- da un carattere attivo; un carattere attivo è un segno che ha ricevuto in precedenza l'etichetta di carattere attivo; LaTeX dichiara carattere attivo il segno `~` che serve per inserire uno spazio non separabile (in corrispondenza del quale non può cadere la fine di una riga); **babel** con l'opzione dell'italiano e di quasi tutte le lingue che usano caratteri accentati, definisce attivo anche il carattere `"`.

Si raccomanda ai compositori di non "giocherellare" con i caratteri attivi; qui li si è nominati perché è giusto sapere che esistono e che cosa sono, ma è meglio lasciarli stare. È meglio lasciare stare anche i comandi che usino caratteri non alfabetici, come per esempio `@`.[1]

Come primo esempio si riporta quanto è stato definito nel preambolo di questo documento, la definizione del comando per scrivere il logo di BibTeX:

```
\newcommand{\BibTeX}{\textsc{Bib}\TeX}
```

Come si vede non ci sono argomenti né obbligatori né facoltativi per la definizione della macro `\BibTeX`; la ⟨*definizione*⟩ si avvale di altri comandi: `\textsc` per comporre la prima parte del logo con il carattere maiuscoletto, e `\TeX` per comporre la seconda parte formata dal logo di TeX.

Come secondo esempio si porta la definizione del comando `\cs` che è servito tante volte per scrivere i nomi dei comandi con carattere a spaziatura fissa e preceduti dal segno di backslash senza che LaTeX li eseguisse, ma li considerasse solo delle stringhe letterali:

[1] Nei file di formato compaiono una quantità di definizioni di comandi che contengono il carattere `@`; all'interno del formato questo carattere è definito come lettera alfabetica, ma fuori del formato, quando il compositore usa LaTeX, esso è ripristinato alla categoria di 'non-lettera'; la cosa è fatta di proposito affinché i compositori non si mettano a giocare con i caratteri non alfabetici.

```
\newcommand*{\cs}[1]{\texttt{\char92#1}}
```

Il comando accetta un solo argomento obbligatorio e la sua sintassi diventa perciò

`\cs{`⟨*nome della macro senza backslash*⟩`}`

La definizione della macro[2], oltre a contenere il comando `\texttt` per comporre il suo argomento con caratteri a spaziatura fissa, contiene anche il comando primitivo `\char` che si aspetta un numero decimale, oppure esadecimale, oppure ottale; dalle tabelle dei caratteri si vede che è più semplice riferirsi all'indirizzo decimale, invece che a quelli ottale o esadecimale, e si vede che il segno di backslash è nella casella con indirizzo 92; usando il comando `\char` e l'indirizzo nella tabella dei caratteri, il segno identificato viene trascritto tale e quale, prescindendo dal significato che esso potrebbe avere per LaTeX, cioè quello di iniziatore di nomi di macro: `\cs` è diverso da `\char92cs` il primo è un comando, il secondo è il nome di un comando.[3]

Nel preambolo di questo documento non sono stati definiti comandi con argomento facoltativo; si prenderà un esempio da altri documenti; prendiamo una possibile definizione del comando `\\`:

```
\newcommand*{\\}[1][0pt]{\newline\vspace{#1}}
```

Questa non è la definizione completa che compare all'interno del file di formato, ma rende l'idea. Il comando `\\` accetta un argomento; siccome ne è specificato il valore di default, questo primo ed unico argomento è facoltativo; se lo si vuole indicare lo si deve scrivere fra parentesi quadre. Se non lo si indica affatto, viene usato il suo valore di default che viene passato tale e quale alla macro `\vspace` che esegue una spaziatura verticale pari all'ammontare specificato con l'argomento esplicito o di default; la spaziatura verticale di default è di 0 pt, cioè una spaziatura nulla.

Il lettore attento avrà notato che in due dei tre esempi un asterisco segue il comando `\newcommand`; questo asterisco facoltativo esprime una variante del comando che è consigliabile usare ogni volta che il comando debba ricevere degli argomenti; senza asterisco il comando può ricevere argomenti arbitrariamente lunghi, fatti anche di numerosi capoversi; con l'asterisco il comando può ricevere argomenti formati solo da una stringa di caratteri che non contenga né esplicitamente né implicitamente una terminazione di capoverso; questa come si ricorda, è costituita da una riga del file sorgente completamente vuota oppure dal comando esplicito `\par`.

Quando si definiscono comandi che accettano solo una o poche parole, certamente non più di un capoverso, è prudente inserire l'asterisco, perché nel caso che ci si dimentichi di scrivere la graffa di chiusura, viene emesso subito

[2] In realtà la definizione indicata è solo una versione preliminare; quando si è deciso di comporre anche l'indice analitico, si è cambiata la definizione includendovi il comando `\index` corredato di un opportuno argomento; la necessità di usare il segno `@`, che ha un significato particolare per makeindex, ci ha obbligati ad una definizione decisamente più complessa, la cui spiegazione esula da questo testo. Tuttavia è importante notare come la modifica della sola definizione di `\cs` ha permesso di inviare all'indice analitico tutti i comandi descritti, senza bisogno di inserire il comando `\index` ogni volta che li si citava.

[3] Invece di `\char92` si sarebbe potuto scrivere `\textbackslash`; ma questo comando agisce solo in modo testo, mentre `\char92` agisce anche in modo matematico. È vero che non lo si è mai usato in modo matematico, ma non si sa mai...

un messaggio d'errore che consente al compositore di intervenire e di correggere. Ricordiamoci che l'omissione di una graffa chiusa rappresenta l'errore più frequente quando si compone il testo sorgente.

Si noti che se si specificasse come nome della ⟨*macro*⟩ quello di un comando già esistente, `\newcommand` protesterebbe con un avviso di errore e non procederebbe ad eseguire la ridefinizione, presumibilmente non desiderata, di un comando già disponibile.

16.3 Ridefinizione di comandi già esistenti

Per ridefinire un comando già esistente, ammesso che il compositore sappia che cosa sta ridefinendo e ne capisca tutte le possibili conseguenze, si usa il comando `\renewcommand` con la sintassi seguente:

```
\renewcommand{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
\renewcommand*{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
```

I cui parametri hanno lo stesso significato che per `\newcommand` e al quale si può aggiungere l'asterisco come si faceva con `\newcommand`.

Il caso più frequente della ridefinizione di comandi esistenti riguarda il caso della modifica di comandi propri oppure di comandi definiti nei file di classe o di estensione.

In questi casi è *vietato* modificare i file originali, ma si procede come indicato nel paragrafo 13.7.

Va da se che se si cerca di ridefinire un comando che non esiste, l'istruzione `\renewcommand` emette un perentorio messaggio d'errore e si rifiuta di procedere oltre.

16.4 Ridefinizioni di comandi di sistema

La ridefinizione dei comandi di sistema è una operazione da evitare se non si sa esattamente quello che si sta facendo; si rischia di buttare all'aria tutto il funzionamento di LaTeX e/o pdfLaTeX. Si veda anche quanto si è scritto nel paragrafo 13.7.

Tuttavia all'inizio si è detto che una delle pochissime modifiche che è stata apportata ai comandi di default per la composizione di questo testo è stata quella di modificare la scrittura dei numeri romani minuscoli. Per eseguire questa scrittura il sistema di default usa esternamente il comando `\roman`, usabile dal compositore, ma internamente il comando del compositore viene eseguito da `\@roman`; il comando comincia con il segno @ che per il compositore non è una lettera e quindi difficilmente il compositore può andare a 'disturbare' i comandi interni di sistema. Tuttavia il modo esiste, ed è descritto da Leslie Lamport in persona nel suo manuale. Bisogna usare il comando `\makeatletter`[4] prima di ridefinire un comando interno; questa istruzione cambia i codici interni che descrivono la funzione del segno @ in modo da poterlo usare come se fosse una lettera dell'alfabeto.

[4] Se la nuova definizione viene inclusa nel proprio file di macro personali, avente estensione `.sty` e richiamato con il comando `\usepackage`, non è necessario servirsi del comando `\makeatletter`.

Questo risolve il primo problema. Ma per scrivere in maiuscoletto non basta specificare `\textsc` perché questa forma non è definita per tutte le famiglie e per tutte le serie. Bisogna accontentarsi di simulare la forma maiuscoletta con lettere maiuscole di corpo più piccolo.

La ridefinizione di `\@roman` diventa allora la seguente:

```
\makeatletter
\DeclareRobustCommand*{\simulatedSC}[1]{%
\hbox{$\relax$\fontsize{\sf@size}{\f@baselineskip}%
                                        \selectfont#1}}}%
\renewcommand*{\@roman}[1]{\simulatedSC{\@Roman#1}}%
```

Una spiegazione però è necessaria. I comandi `\sf@size` e `\f@baselineskip` sono i comandi interni che contengono rispettivamente il corpo degli indici primi e dell'avanzamento di riga associati al font corrente; il primo dei due è definito solo dopo che sia stata scritta una formula, qui ridotta a nulla (`\relax`); essi sono documentati insieme ad altri comandi dello stesso genere nella già citata guida `fntguide` presente in `.../doc/latex/base/`. Il comando `\fontsize` è quello esterno per impostare un nuovo corpo e un nuovo avanzamento di riga; ma questi non diventano operativi se non dopo aver dato il comando esplicito `\selectfont`; infine il comando `\@Roman` è il comando interno per scrivere i numeri romani in lettere maiuscole. Ciò premesso la ridefinizione che stiamo cercando di fare richiede un comando `\simulatedSC` con il quale si ordina di scrivere con l'avanzamento di riga corrente, ma con il corpo degli indici primi; per limitare i suoi effetti e per essere sicuri che il numero romano, cadendo in fin di riga, non venga eventualmente divisio in sillabe, il tutto viene eseguito dentro una scatola orizzontale `\hbox` che successivamente viene usata come un unico oggetto, non come una stringa di lettere/cifre romane; usando questo comando la ridefinizione di `\@roman` viene eseguita in modo da prendere il numero romano scritto in lettere maiuscole e di stamparlo nel corpo degli indici primi. Il risultato è quello che si può vedere nelle pagine iniziali; qui si mostra come la pagina corrente possa essere scritta in numeri romani maiuscoli e in numeri romani maiuscoletti in corpi diversi:

Corpo 'normalsize': CLXXXIII CLXXXIII

Corpo 'Huge': CLXXXIII CLXXXIII

Un altro esempio usato in questo testo è costituito dalla ridefinizione della virgola. È noto che in tutte le lingue, tranne l'inglese, le norme ISO prescrivono come separatore decimale la virgola. In questo modo in tutte le altre lingue la virgola svolge due ruoli, ma solo in matematica. Quando si è in modo matematico il segno della virgola può essere dichiarato attivo e associargli una definizione. Qui si è fatto esattamente così:

```
\DeclareMathSymbol{\virgola}{\mathpunct}{letters}{"3B}
\DeclareMathSymbol{\decimalcomma}{\mathord}{letters}{"3B}
\AtBeginDocument{\mathcode`\,="8000}
{\catcode `,=\active \gdef,{\futurelet\let@token\m@thcomma}}
\def\m@thcomma{%
\ifx\let@token\@sptoken
    \virgola
```

```
\else
    \decimalcomma
\fi}
```

Precisamente si sono definite due entità matematiche: `\virgola` che rappresenta un segno matematico di punteggiatura e `\decimalcomma` che rappresenta la virgola decimale; la differenza risiede nel codice interno che il carattere `"3B` riceve attraverso i comandi `\DeclareMathSymbol`, visto che nel primo caso è specificato appartenente alla categoria `\mathpunct` (punteggiatura matematica) mentre nel secondo caso esso è definito come `\mathord`, cioè come un normale simbolo o una variabile matematica. La specifica `letters` in entrambi i casi dice che il segno deve essere tratto dalla stessa polizza di caratteri dai quali si traggono le lettere (il font corsivo matematico).

Le definizioni del codice attivo e della sua definizione diventano attive solo all'inizio del documento mediante `\AtBeginDocument`; vi si dice che il codice matematico deve essere posto a `"8000`, lo speciale codice che definisce attivo un carattere in modo matematico. Gli si assegna poi attraverso il comando primitivo `\gdef` una definizione globale nella quale esso deve assegnare a `\let@token` non il significato del token successivo, che sarebbe `\m@comma`, ma quello del token che incontrerà ancora dopo; allora e solo allora potrà eseguire `\m@comma`. Questo a sua volta è definito in modo da esaminare il tipo di token il cui significato è stato attribuito a `\let@token`; se questo è uno spazio (`\@sptoken`) usa la virgola di punteggiatura, altrimenti usa la virgola come simbolo. I comandi `\ifx`, `\else` e `\fi` servono appunto per eseguire questi comandi condizionali. Sta ora al compositore lasciare uno spazio dopo la virgola (in matematica) solo quando questa rappresenta un segno di punteggiatura, mentre non deve lasciare nessuno spazio se deve comportarsi da separatore decimale.

Il risultato di questo comando è visibile in ogni numero decimale fratto scritto in questo testo, ma qui è opportuno confrontare bene che cosa si ottiene se in matematica si scrive `f(x,y)` oppure `f(x,␣y)`: $f(x,y)$ nel primo caso, ma $f(x, y)$ nel secondo. La differenza è piccola, ma si vede benissimo.

Componendo in inglese non si deve eseguire nessuna definizione strana della virgola; quindi se si vogliono inserire questi comandi in un file di macro personali, è meglio racchiuderli dentro gli argomenti del comando di **babel** `\iflanguage` che ha la seguente sintassi:

`\iflanguage{`⟨*lingua*⟩`}{`⟨*vero*⟩`}{`⟨*falso*⟩`}`

dove lingua è il nome dell'opzione passata a **babel** come lingua di default, mentre ⟨*vero*⟩ sono le azioni da eseguire se la lingua di default è quella specificata, altrimenti si eseguono le azioni indicate da ⟨*falso*⟩.

Nel nostro caso converrebbe inserire nell'argomento di `\AtBeginDocument` il costrutto:

```
\AtBeginDocument{\iflanguage{english}{\relax}{\mathcode`\,="8000}}
```

cioè: se la lingua di default è l'inglese, rilassati (non fare nulla), altrimenti dichiara attivo il simbolo della virgola in matematica.

Questi due esempi ricorrono a comandi primitivi e ai codici di categoria matematici; non si tratta di concetti adatti ad una introduzione, ma non sono nemmeno concetti inaccessibili; il lettore che voglia approfondire deve esaminare e studiare il TeXbook, dove questi argomenti sono trattati in diversi capitoli; si veda nell'appendice A.

16.5 Esiste già o non esiste ancora il comando?

Talvolta si vuole essere sicuri che un comando sia disponibile ma non si sa se esso sia già stato definito in qualche pacchetto di estensione. Allora si può usare il comando `\providecommand` che ha la stessa sintassi di `\newcommand`

```
\providecommand{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
\providecommand*{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
```

e accetta anche l'asterisco facoltativo con lo stesso significato che esso ha per `\newcommand`.

Si ricorre spesso a questo tipo di definizione nel proprio file di macro personali, cosicché il comando definito attraverso questa istruzione viene sempre definito se il comando non è già stato definito da altri, altrimenti questa definizione viene tranquillamente ignorata e viene usata la definizione già esistente. Logicamente perché questo funzioni come desiderato, è necessario che il proprio file di macro personali sia sempre invocato per ultimo.

Nel preambolo di questo testo, per esempio, si è provveduto il comando `\ohm` nel modo seguente

```
\providecommand*{\ohm}{\textormath{\textohm}{\mathrm{\Omega}}}
```

dove `\textormath` è un comando fornito dal pacchetto **babel** per specificare due azioni diverse da eseguire in modo testo invece che in modo matematico; in modo matematico si è specificato di usare il font tondo diritto, per ovviare al caso in cui si sia deciso di usare le lettere greche maiuscole inclinate nelle normali espressioni matematiche.

16.6 Definizione di comandi robusti

Un comando si dice robusto se esegue quel che deve eseguire in qualunque circostanza; ciò non è sempre vero, specialmente se esso contiene nella sua definizione dei comandi condizionali e se costituisce o può costituire l'argomento di un altro comando.

È prudente, allora, definire comandi robusti confezionati con cautele particolari, i cui dettagli sono troppo tecnici da spiegare qui. Il tutto viene eseguito mediante il comando `\DeclareRobustCommand` che può venire usato solo nel preambolo o, bene inteso, anche nei file di classe o di estensione, che comunque vengono letti solo nel preambolo. La sua sintassi è identica a quella di `\newcommand`; bisogna stare attenti che esso non verifica se il comando da definire esista già o sia davvero un comando nuovo, quindi questa 'dichiarazione' deve venire usata con molta cautela da programmatori esperti; tuttavia se si fosse definito un comando con `\newcommand` e questo avesse passato i controlli di definibilità, ma nell'uso risultasse fragile, allora non sarebbe un problema cambiare la dichiarazione `\newcommand` con `\DeclareRobustCommand` e tutto procederebbe come desiderato con la certezza di non avere pasticciato con comandi già esistenti.

Nell'appendice E viene descritto il comando `\protect`; esso serve per rendere robusti i comandi fragili; il comando `\DeclareRobustCommand` agisce attraverso `\protect` ma passa anche per l'intermediario di un nuovo comando, apparentemente con lo stesso nome di quello che costituisce il suo argomento,

ma prolungato con uno spazio che fa parte integrante del suo nome: se si vuole rendere robusto il comando `\pippo`, `\DeclareRobustCommand` protegge mediante `\protect` il comando `\pippo␣` e a questo assegna la definizione specificata. Questo aiuta a rendere robusti i comandi quando essi vengono scritti nei file ausiliari; quando questi file vengono riletti in una esecuzione successiva, tutto quello che viene letto ripassa attraverso il processo di 'tokenizzazione', e lo spazio letto dal file ridiventa un normale spazio privo di significato quando segue il nome di un comando. Ingegnoso, ma talvolta i neofiti non riescono a rendersi conto di questo passaggio.

16.7 Definizione di un nuovo ambiente

Per definire un nuovo ambiente si ricorre al comando `\newenvironment` con la seguente sintassi:

```
\newenvironment{⟨ambiente⟩}[⟨num_arg⟩][⟨default⟩]%
              {⟨apertura⟩}{⟨chiusura⟩}
```

L'⟨*ambiente*⟩ è il nome dell'ambiente che si vuole definire e che non deve esistere già, altrimenti viene emesso un messaggio d'errore e la compilazione del documento viene bloccata.

L'ambiente che viene definito può ricevere *solo in apertura* un certo numero ⟨*num_arg*⟩ di argomenti il primo dei quali può essere facoltativo, se viene specificato ⟨*default*⟩ e va trattato come il primo argomento facoltativo definito con `\newcommand`.

Seguono poi i comandi da eseguire in ⟨*apertura*⟩ dell'ambiente, il quali fanno uso esplicito degli argomenti eventualmente presenti, obbligatori o facoltativi; infine vengono definiti i comandi da usare in ⟨*chiusura*⟩ dell'ambiente.

In realtà definire un ⟨*ambiente*⟩ equivale a definire due comandi uno per l'⟨*apertura*⟩ e l'altro per la ⟨*chiusura*⟩, come se si fossero definiti i due comandi seguenti

```
\newcommand{\⟨ambiente⟩}[⟨num_arg⟩][⟨default⟩]{⟨apertura⟩}
\newcommand{\end⟨ambiente⟩}{⟨chiusura⟩}
```

I comandi `\begin` e `\end` con i quali si aprono e si chiudono gli ambienti eseguono delle funzioni importanti fra le quali la costituzione di un gruppo, cosicché tutto quello che si fa dentro all'ambiente rimane locale e circoscritto solo e soltanto a quell'ambiente; il comando `\end`, oltre a chiudere il gruppo, provvede a verificare che si stia chiudendo l'ultimo ambiente che si era aperto; in caso contrario viene emesso un messaggio d'errore.

Il fatto dei due comandi distinti per l'apertura e la chiusura di un ambiente può essere sfruttato nella definizione di nuovi ambienti; per esempio in questo documento il preambolo definisce l'ambiente *sintassi* che provvede a stampare la sintassi dei vari comandi incorniciando il suo contenuto in un rettangolo che fa da cornice. In questo modo si è pensato di definire l'ambiente *medaglione* che incornicia il suo contenuto e l'ambiente *sintassi* che espone il medaglione incorniciato ma separato dal testo che lo precede e che lo segue di uno spazio verticale uguale a quello che si ha nei testi in display; in particolare si richiede che il testo sia giustificato solo a sinistra; le due definizioni sono le seguenti:

```
\newenvironment{medaglione}[1][\linewidth]{\setbox0\vbox\bgroup
       \hsize#1\advance\hsize-2\fboxsep\advance\hsize-2\fboxrule
\noindent}{\par\egroup\setbox0\vbox{\unvbox0}%
       \framebox{\box0}}
%
\newenvironment{sintassi}{\flushleft\medaglione}
                    {\endmedaglione\endflushleft}
```

L'interpretazione delle definizioni per l'ambiente *sintassi* è abbastanza semplice: in apertura esegui le istruzioni per comporre un testo in display giustificato solo a sinistra e poi esegui le operazioni per l'apertura di un medaglione; in chiusura esegui le operazioni per terminare il medaglione e quelle per terminare il testo in display giustificato a sinistra.

I comandi per la definizione del *medaglione* sono più complicati e richiedono la conoscenza approfondita dei comandi primitivi. Si osservi innanzi tutto che l'ambiente accetta un argomento facoltativo il cui valore di ⟨*default*⟩ vale `\linewidth`; questo argomento facoltativo è stato usato per comporre, per esempio, il medaglione della pagina 3.

A parte l'argomento facoltativo, in sostanza i comandi di apertura dicono che con `\setbox` si deve riempire un registro di tipo `box`, quello numerato 'zero' con una scatola verticale `\vbox` la cui apertura è indicata con il comando `\bgroup`; in questa scatola verticale compone del testo usando una giustezza di `\hsize` impostato al valore corrente del primo argomento `#1` esplicitamente specificato o il suo valore di default; ma si diminuisce questo valore di quanto basta per tenere conto dello spazio fra la cornice e il suo contenuto, valore che è conservato dentro il registro di tipo 'lunghezza' chiamato `\fboxsep`, e dello spessore del filetto conservato dentro il registro `\fboxrule`; inoltre si ordina di comporre senza rientro di capoverso mediante `\noindent`.

Nei comandi di chiusura bisogna accertarsi che il capoverso costituito dal contenuto del medaglione sia effettivamente terminato ordinandone la terminazione con il comando `\par` e si termina il gruppo che chiude la scatola verticale con `\egroup`; si riassegna poi il contenuto della scatola 'zero' ancora alla scatola 'zero' estraendo le singole righe ovverosia disfacendo la scatola 'zero' con il comando `\unvbox`; questo serve per essere sicuri che le righe siano convenientemente spaziate; spesso questa operazione è inutile, ma non guasta; infine si usa questa scatola con `\box` come argomento di `\framebox` al fine di incorniciarla.

La persona che si avvicina per la prima volta al contenuto di questo capitolo può restare sconcertata anche da questo linguaggio strano, costituito dai comandi primitivi di TEX che, tra l'altro, non sono mai citati o descritti in questo testo. Be', questo testo è introduttivo, non esaustivo; il lettore si accontenti di questo primo assaggio e poi si documenti convenientemente nel TEXbook se vuole andare oltre a questi primi passi.

Nello stesso tempo si capisce bene a che cosa possano servire le macro; per eseguire nell'ordine giusto tutta una serie di operazioni; se si dovessero specificare ogni volta che di deve fare una certa operazione costringerebbero a scrivere e a riscrivere sempre le stesse cose con il pericolo di aggiungere errori ogni volta che si ripetono queste scritture.

16.8 La ridefinizione di ambienti esistenti

La ridefinizione di ambienti già esistenti si esegue con `\renewenvironment` che usa la stessa sintassi di `\newenvironment`. Viene scambiato solo il tipo di test; se l'ambiente da ridefinire non esiste, viene emesso un messaggio di errore e la compilazione viene interrotta.

Per gli ambienti non esiste un comando simile a `\providecommand`, ma in base a ciò che è stato detto a proposito dei nuovi ambienti non dovrebbe essere difficile per chiunque usare due volte il comando `\providecommand` per definire l'apertura e la chiusura di un ambiente; purché tutto ciò sia fatto nel proprio file di macro personali.

Non è nemmeno difficile copiare dai file di classe o di estensione le definizioni di ambienti già esistenti travasando le definizioni nel proprio file di macro personali, per poi separare le due definizioni di apertura e di chiusura in due distinti comandi `\providecommand` dentro alle cui definizioni apportare tutte le modifiche che si credono opportune.

Qui si porterà un esempio abbastanza articolato relativo alla modifica di un ambiente di sistema, la modifica dell'ambiente *theindex* per la composizione dell'indice analitico.

Si vorrebbe cambiare la formattazione nel senso di comporre l'indice sempre a due colonne, ma facendolo precedere da una spiegazione per l'uso composta a piena pagina. Vorremmo anche che l'indice analitico figurasse anche nell'indice generale. La prima cosa da fare è quella di esplorare l'archivio CTAN per vedere se il problema sia già stato risolto. Qui supporremo di non aver trovato nulla che vada bene per il nostro caso e quindi, rimboccandoci le maniche, ma esercitando la nostra creatività, ci accingiamo a modificare la definizione esistente.

Bisogna quindi copiare la definizione dell'ambiente *theindex* dal file di classe che stiamo usando per comporre il nostro documento nel nostro file di macro personali `mymacros.sty`. Per la classe *book* copieremo pertanto il codice seguente:

```
\newenvironment{theindex}
               {\if@twocolumn
                  \@restonecolfalse
                \else
                  \@restonecoltrue
                \fi
                \twocolumn[\@makeschapterhead{\indexname}]%
                \@mkboth{\MakeUppercase\indexname}%
                        {\MakeUppercase\indexname}%
                \thispagestyle{plain}\parindent\z@
                \parskip\z@ \@plus .3\p@\relax
                \columnseprule \z@
                \columnsep 35\p@
                \let\item\@idxitem}
               {\if@restonecol\onecolumn\else\clearpage\fi}
\newcommand\@idxitem{\par\hangindent 40\p@}
%
\newcommand\subitem{\@idxitem \hspace*{20\p@}}
\newcommand\subsubitem{\@idxitem \hspace*{30\p@}}
```

```
\newcommand\indexspace{\par
                \vskip 10\p@ \@plus5\p@ \@minus3\p@\relax}
```

Le ultime definizioni servono per formattare le singole voci, in particolare per inserire le debite indentature per mettere in evidenza i tre livelli di voci disponibili. L'ultima definizione, quella di `\indexspace` serve per inserire uno spazio verticale nell'elenco composto ogni volta che si inizia una serie di voci con una nuova lettera dell'alfabeto. Volendo quest'ultimo comando potrebbe venire modificato per poter non solo lasciare lo spazio ma anche per scrivere la nuova lettera dell'alfabeto.

Per poter scrivere su due colonne dopo aver cominciato a scrivere a piena pagina è conveniente aver caricato il pacchetto **multicol**; potremmo anche inserire un test per verificare se il pacchetto è stato davvero già caricato, ma per semplicità ometteremo questo controllo.

Allora modifichiamo la definizione dell'ambiente, in particolare la definizione dei comandi di apertura: (*a*) in modo da inserire l'ordine di scrivere nell'indice generale il nome dell'indice analitico e in modo da non usare il comando `\twocolumn` ma l'ambiente *multicols* fornito dal pacchetto **multicol**. Lasciamo stare gli ultimi tre comandi come sono, per cui qui non li ripetiamo. Ovviamente cambiamo `\newenvironment` in `\renewenvironment` visto che stiamo ridefinendo un ambiente esistente.

```
\renewenvironment{theindex}
     {\if@twocolumn
         \@restonecolfalse
       \else
          \@restonecoltrue
        \fi
        \@makeschapterhead{\indexname}%
        \@mkboth{\MakeUppercase\indexname}%
                {\MakeUppercase\indexname}%
        \addcontentsline{toc}{chapter}{\indexname}%
        \thispagestyle{plain}%
        \parindent\z@
        \parskip\z@ \@plus .3\p@\relax
        \columnseprule \z@
        \columnsep 35\p@
        \let\item\@idxitem
        %               Testo di spiegazione
        In questo indice analitico le voci hanno
        il numero di pagina in neretto quando si
        tratta di una definizione,  e in corsivo
        quando si tratta di un esempio d'uso.
        \par
        \vspacs{3\baselineskip}%
        \multicols{2}}% Fine comandi di apertura
       {%               Inizio comandi di chiusura
        \endmulticols
        \if@restonecol\onecolumn\else\clearpage\fi}
```

Qualche parola di spiegazione non guasta. Il test iniziale eseguito mediante il comando `\if@twocolumn` appare all'inizio di ogni definizione di comandi o di

ambienti che comincino un nuovo capitolo numerato o non numerato. Servono per stabilire se dopo la chiusura dell'ambiente sia o non sia necessario ripristinare la composizione a una colonna; l'ultimo test, nei comandi di chiusura eseguito con il comando `\if@restonecol`, serve appunto per ripristinare la composizione ad una colonna nel caso che sia necessario.

Il comando `\@makeschaperhead` è il comando con il quale viene composto il titolo di un capitolo non numerato; in questo caso il titolo del capitolo è contenuto nella variabile `\indexname` che **babel** carica con il nome giusto a seconda della lingua in uso: 'Indice analitico', 'Index',...

Il comando `\mkboth`, seguito dai suoi due argomenti serve per impostare il contenuto delle due testatine, prima viene indicato il contenuto della testatina di sinistra, poi quello di destra. Il contenuto è sempre il nome in lingua corrente dell'indice analitico, ma reso in lettere maiuscole attraverso il comando `\MakeUppercase`.

Il comando `\addcontentsline`, come dice il nome, serve per aggiungere una riga alla 'table of contents', cioè all'indice generale. I suoi tre argomenti indicano rispettivamente la sigla della 'table of contents' `toc`, il tipo di voce da aggiungere a questa lista; in questo caso si tratta di una voce corrispondente al titolo di un capitolo; segue poi il titolo del capitolo che in questo caso continua ad essere il nome dell'indice analitico nella lingua corrente.

Il comando `\thispagestyle` specifica lo stile compositivo di questa prima pagina di questo nuovo capitolo; in questo caso si tratta dello stile `plain` che non contiene la testatina, ma contiene il numero della pagina nel piedino. Gli altri stili definiti di default sono `empty`, senza testatina né piedino, `headings` con piedino vuoto e testatina il cui contenuto viene specificato automaticamente dai comandi `\chapter` oppure `\section`; in particolare nella testatina di sinistra compare il titolo del capitolo e in quella di destra il titolo del paragrafo corrente all'inizio della pagina. Infine `myheadings` è simile a `headings` solo che il contenuto delle due testatine deve essere fissato manualmente dal compositore specificando di volta in volta entrambe le testatine oppure solo quella di destra attraverso i comandi `\markboth` oppure `\markright`.

La scrittura `\parindent \z@` serve per specificare che il rientro del capoverso deve essere di zero punti; questo valore è memorizzato nella variabile di sistema `\z@` che equivale alla scrittura 0 pt.

La successiva espressione `\parskip\z@ \@plus .3\p@\relax` serve per specificare un pochino di gomma elastica da inserire fra un capoverso e l'altro; in pratica fra una voce e l'altra; questo minimo di allungamento consente di giustificare verticalmente le colonne della composizione a due colonne.

La specificazione successiva `\columnseprule \z@` specifica che lo spessore del filetto verticale che separa le colonne è di zero punti (quindi il filetto non c'è). Invece `\columnsep 35\p@` dice che la separazione fra le due colonne vale 35 pt, quindi una decina di millimetri.

Al comando `\item`, che abitualmente viene usato nelle liste, qui viene assegnata un'altra definizione, quella che è stata attribuita al comando `\@idxitem` e che non è stata modificata nell'eseguire questa nuova definizione dell'ambiente *theindex*.

Segue poi il testo di spiegazione, che qui è stato limitato a poche righe, giusto per fare un esempio; alternativamente questo testo potrebbe essere incluso in una macro e, invece di usare il testo completo nella definizione del nuovo ambiente, si sarebbe potuto indicare solo il nome della macro.

Finita la spiegazione il comando `\par` assicura che il capoverso contenente le spiegazioni sia veramente terminato; il successivo comando di spaziatura verticale dato con `\vspace` inserisce una spaziatura di tre righe di testo; infatti `\baselineskip` indica appunto l'avanzamento di riga, quindi l'altezza completa di interlinea di una riga di testo.

I comandi di apertura finiscono con l'apertura dell'ambiente *multicols* al quale viene specificato di comporre in due colonne; non è stata usata la sintassi normale per aprire l'ambiente, perché siamo sicuri di chiuderlo nei comandi di chiusura, come in effetti facciamo eseguendo il comando `\endmulticols`.

Questo esempio è istruttivo sotto molti aspetti; dà una buona idea di ciò che viene eseguito dietro le quinte quando si dà un comando qualunque; dice quanto sia comodo che le macro posano fare tutto quello che fanno, così che con un semplice comando di mark-up si possano sistematicamente e uniformemente eseguire le stesse operazioni compositive; quanto lavoro si risparmi nell'usare macro ogni volta che sia possibile, e quindi quanto sia opportuno che, in base al testo che si sta componendo, il file `mymacros.sty` contenga proprio le definizioni delle macro personali che servono per comporre le strutture lessicali che più frequentemente compaiono nel testo.

16.9 Immagini, celle e scatole

Il titolo di questo paragrafo è un po' enigmatico; in realtà il problema è semplice: inserire immagini nelle celle di una tabella può risultare difficile, a meno che non si sia disposti a manipolare le scatole.

Nel resto di questo testo non si è detto molto sulle scatole; l'arte di eseguire la composizione tipografica con LaTeX dovrebbe astenersi dal manipolare le scatole interne dell'interprete TeX, almeno in prima battuta.

Ciò non toglie che qualche nozione non possa essere utile, anche per capire meglio alcuni strani costrutti eseguiti con le scatole nella definizione di alcuni comandi.

TeX, e quindi LaTeX e pdfLaTeX, maneggia quasi esclusivamente scatole; sono scatole i singoli caratteri, le righe del testo, le pagine composte, tanto per fare esempi molto semplici. Ogni scatola è definita dal suo tipo e da tre dimensioni particolari: la larghezza, l'altezza e la profondità. Il rettangolo che può simboleggiare una scatola ha un punto di riferimento all'intersezione del bordo sinistro con la linea che separa la sua parte alta dalla sua parte profonda; il segmento intercettato su questa linea dal bordo destro e dal bordo sinistro è lungo quanto la larghezza della scatola; l'altezza è la lunghezza del segmento che parte dal punto di riferimento e raggiunge il vertice in alto a sinistra del rettangolo, mentre la profondità è la lunghezza del segmento che parte dal punto di riferimento e raggiunge il vertice in basso a sinistra; figura 16.1.

Le scatole dei caratteri vengono allineate a formare una riga semplicemente allineando i loro punti di riferimento sulla linea di base della riga composta, come si vede nella figura 16.1; TeX prende le dimensioni delle scatole che racchiudono i caratteri dal file metrico del font che sta usando; questo contiene non solo tutte le informazioni dimensionali di ciascun carattere, ma anche le dimensioni delle crenature[5] e le possibili legature fra caratteri. La riga così formata rappresenta

[5]La crenatura (*kerning* in inglese) è l'ammontare dell'accostamento di due caratteri adiacenti, in modo da non fare apparire troppo spazio fra caratteri consecutivi dalle forme com-

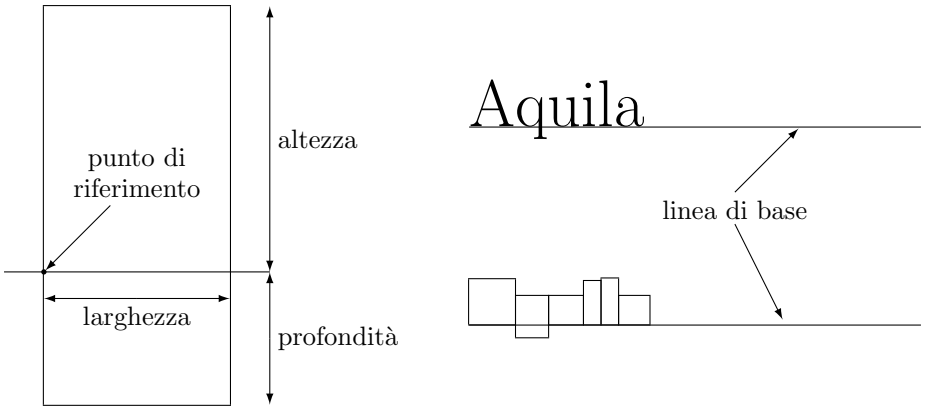

Figura 16.1: Alcune proprietà delle scatole. Nella figura a destra si vedono le scatole corrispondenti ai caratteri che formano la parola 'Aquila' allineate sulla linea di base. Per la verità la sequenza di scatoline è più lunga della parola che esse rappresentano, perché non si è tenuto conto delle crenature fra i caratteri.

una scatola orizzontale, la cui larghezza è la somma delle larghezze dei caratteri componenti e degli spazi bianchi che separano le parole; l'altezza della riga è la maggiore delle altezze delle scatole allineate orizzontalmente, mentre la profondità della riga è la maggiore delle profondità delle scatole suddette; il punto di riferimento è perciò l'intersezione della linea di base con il lato sinistro del rettangolo che racchiude la riga composta.

Le scatole formate dalle righe composte vengono incolonnate in verticale allineando i punti di riferimento di tutte le scatole orizzontali usate lungo una retta verticale e intercalando alle scatole orizzontali gli spazi bianchi di interlinea e i contrografismi verticali prima e dopo ogni scatola che contenga del testo in evidenza. La pagina composta è quindi una scatola verticale; ma TeX può gestire anche altre scatole, differenti dall'intera pagina, scatole che a loro volta possono contenere altre scatole, e così via come una serie di matriosche.

LaTeX ha due tipi di scatole orizzontali che l'utente può usare con comandi accessibili direttamente dal compositore: `\mbox` e `\makebox`; alcune varianti sono le scatole con cornice, come `\fbox` e `\framebox`; il compositore può definire per nome dei registri-scatola nei quali può salvare scatole orizzontali mediante i comandi `\newsavebox`, `\sbox` e `\savebox`; il loro contenuto viene richiamato e usato con `\usebox`.

LaTeX ha due tipi di scatole verticali a cui il compositore può accedere liberamente; una è la scatola verticale prodotta con il comando `\parbox` e l'altra è la 'minipagina' ottenibile con l'ambiente *minipage*. Le scatole verticali di LaTeX possono ricevere il loro contenuto come sequenze di caratteri e di altri oggetti tipicamente di tipo orizzontale, oppure come scatole orizzontali già composte; nel primo caso LaTeX provvede a costruire le scatole orizzontali da impilare in verticale. La larghezza di simili scatole verticali coincide con quella della scatola orizzontale più larga, l'altezza e la profondità dipendono dalle opzioni che vengono specificate al comando o all'ambiente, dipendendo da come queste op-

plementari; per esempio 'VA' sono accostate, mentre 'VA' non lo sono; la differenza è evidente e un font nel quale manchi l'accostamento sarebbe di qualità assai modesta.

zioni specificano la posizione del punto di riferimento. La scatola di tre righe Pippo può avere il suo punto di riferimento collocato in modo che la prima Pluto
Paperino

Pippo
Pluto
riga sia allineata con il testo circostante, oppure Paperino può avere l'ultima
Pippo
riga allineata con il testo circostante; oppure Pluto può avere il suo asse
Paperino
matematico allineato con l'asse matematico del testo circostante. Come si vede da questo capoverso le scatole verticali non sono molto utili da usare all'interno del testo; esse trovano applicazioni particolari in altri tipi di composizione.

I comandi primitivi di TEX sono meno elaborati; i registri-scatola possono essere chiamati per nome, come in LATEX, oppure per numero, purché questo sia strettamente compreso nell'intervallo da 0 a 254; il registro-scatola 255 è riservato e l'utente non lo deve mai toccare; le scatole orizzontali sono contenute dentro `\hbox`; le scatole verticali si chiamano `\vbox`, `\vtop` e `\vcenter`; i comandi per utilizzare il contenuto delle varie scatole registrate nei registri-scatola sono invece più elaborati di quelli di LATEX. Il lettore è invitato ad approfondire l'argomento nel TEXbook, che dedica a questi argomenti diversi capitoli.

Dopo queste premesse, possiamo tornare alle nostre immagini e alle caselle, per esempio delle tabelle; abbiamo visto che uno dei descrittori delle colonne è `p{`⟨*larghezza*⟩`}`; questo in sostanza definisce una scatola verticale di larghezza pari a ⟨*larghezza*⟩ dentro la quale viene composto un breve capoverso; ma questa scatola ha la *prima* riga allineata con la prima o unica riga delle celle adiacenti. Mediante l'uso del pacchetto **array** abbiamo visto che si possono usare altri due descrittori, `b{`⟨*larghezza*⟩`}` e `m{`⟨*larghezza*⟩`}` che si comportano rispettivamente come scatole verticali con l'ultima riga allineata ovvero con l'asse matematico allineato.

Le immagini introdotte nel documento ricorrendo al pacchetto **graphicx** mediante il suo comando `\includegraphics`, sono delle scatole orizzontali col loro punto di riferimento coincidente con il vertice inferiore sinistro. Si capisce allora che per collocare una immagine bisogna tenere conto di altezza e profondità delle scatole usate e regolarsi in modo da ottenere un buon risultato. La tabella 16.1 che si ottiene senza pensare a queste cose è orrenda. La tabella 16.2, composta ricordando le proprietà delle scatole appena descritte, invece, è composta abbastanza correttamente.

La tabella 16.2 può ancora essere ritoccata aggiungendo un piccolo spazio sopra l'immagine, in modo che il filetto orizzontale non la sfiori: tabella 16.3.

La tabella 16.3 è stata composta con i comandi seguenti:

```
\begin{tabular}{m{120pt}m{35mm}}
   \hline
   Immagine & Descrizione\\
   \hline
   \vspace*{.8ex}
   \includegraphics[width=120pt]{Formaggio} &
   Il formaggio è un prodotto derivato dal latte, attraverso un
   processo che coagula la caseina, una proteina del latte.\\
   \hline
```

Immagine	Descrizione
	Il formaggio è un prodotto derivato dal latte, attraverso un processo che coagula la caseina, una proteina del latte.

Tabella 16.1: Tabella in cui l'immagine e il testo adiacente hanno i punti di riferimento allineati in modo scorretto

Immagine	Descrizione
	Il formaggio è un prodotto derivato dal latte, attraverso un processo che coagula la caseina, una proteina del latte.

Tabella 16.2: Tabella in cui l'immagine e il testo adiacente hanno i punti di riferimento allineati correttamente

Immagine	Descrizione
	Il formaggio è un prodotto derivato dal latte, attraverso un processo che coagula la caseina, una proteina del latte.

Tabella 16.3: Tabella in cui l'immagine e il testo adiacente hanno i punti di riferimento allineati correttamente e dove uno spazio verticale pari a 0,8 ex separa l'immagine dal filetto sovrastante

```
\end{tabular}
```

Premesso che il descrittore di colonna `m{`⟨*larghezza*⟩`}` del pacchetto **array** risolve il problema, che cosa bisogna fare se per qualche motivo non si potesse conservare quello stesso tipo di descrizione per tutte le celle della stessa colonna? Bisogna ricorrere a comandi di basso livello, cioè ai comandi che manipolano direttamente le scatole in modo da risolvere il problema compositivo solo per la cella specifica che contiene l'immagine. Conviene anzi definirsi un comando che esegua direttamente l'inclusione dell'immagine, ma con la posizione del punto di riferimento specificata mediante una opzione. Si potrebbe definire un nuovo comando `\tabularimage` con la sintassi:

`\tabularimage[`⟨*posizione*⟩`]{`⟨*larghezza*⟩`}{`⟨*file immagine*⟩`}`

nel modo seguente:

```
\newcommand*\ttoprule{\hrule \@height0pt \@width0pt \@depth-1ex}
\newcommand*\ctoprule{\hrule \@height0pt \@width0pt \@depth.8ex}
\let\btoprule\ctoprule
\let\bbotrule\ctoprule
\let\cbotrule\bbotrule
\let\tbotrule\bbotrule
\newcommand*{\tabularimage}[3][b]{\parbox[#1]{#2}{%
   \csname#1toprule\endcsname %
   \includegraphics[width=#2]{#3}\par
   \csname#1botrule\endcsname}}
```

dove la ⟨*posizione*⟩ di default è la stessa che si avrebbe con il semplice comando `\includegraphics`. Con questo nuovo comando si potrebbe comporre la tabella 16.4 con i comandi:

```
\begin{tabular}{lp{35mm}}
   \hline
   Immagine & Descrizione\\
   \hline
   \tabularimage[t]{120pt}{Formaggio} &
   Il formaggio è un prodotto derivato dal latte, attraverso un
   processo che coagula la caseina, una proteina del latte.\\
   \hline
\end{tabular}
```

La definizione di `\tabularimage` è un po' particolare, perché i filetti (invisibili) inseriti prima e dopo l'immagine all'interno della scatola verticale `\parbox` non possono essere i comuni filetti di LaTeX specificati con `\rule`; infatti qui abbiamo bisogno di incolonnare solo scatole verticali o altri oggetti che si possono mettere in una lista verticale; il filetto `\hrule` nonostante l'iniziale `h`, è un comando primitivo che specifica un filetto le cui altezza, larghezza e profondità sono specificate con i parametri che seguono quel comando, ma è un oggetto che può stare solo nelle liste di oggetti incolonnati in verticale. I punti di riferimento di questi filetti diventano quelli della scatola verticale quando le si specifichi l'opzione di ⟨*posizione*⟩ `t` oppure `b`; nel caso dell'opzione `c`, non conta tanto il punto di riferimento, quanto l'asse matematico, che TeX determina come linea

Immagine	Descrizione
	Il formaggio è un prodotto derivato dal latte, attraverso un processo che coagula la caseina, una proteina del latte.

Tabella 16.4: Tabella con l'immagine allineata con la parte superiore delle minuscole della prima riga della seconda colonna

mediana della scatola. Si capisce, allora, che a seconda che il collocamento della scatola corrisponda all'una delle tre specificabili, i filetti (invisibili) superiore e inferiore debbano avere caratteristiche diverse; si sono definiti sei filetti invisibili, che in realtà sono solo due distinti ma hanno sei nomi che si equivalgono; questi nomi sono ottenuti agglutinando la lettera di opzione per la collocazione con i nomi `toprule` e `botrule`; questa agglutinazione viene eseguita dalla definizione di `\tabularimage` solo al momento in cui il parametro `#1`, che corrisponde a questa opzione, è noto e l'operazione viene eseguita dalla coppia di comandi `\csname` e `\endcsname` che creano un nome di comando valido per TeX a partire da qualsiasi stringa essi racchiudano.

Questo genere di incapsulamenti di scatole con diversi punti di riferimento può essere fatto anche con le minipagine create con l'ambiente *minipage* o con le scatole orizzontali con diversi allineamenti del loro contenuto e diverse larghezze nominali.

Non si insiste oltre essenzialmente per non intimorire l'utente di LaTeX alle prime armi. È certo che la definizione di questi comandi, specialmente ricorrendo ai trucchi esposti, rientra nella programmazione col linguaggio LaTeX, certamente non nell'arte di comporre tipograficamente con LaTeX, anche se ne costituisce il supporto. Ma l'appetito vien mangiando; prima o poi tutti sentiamo il bisogno di definirci i nostri comandi, i nostri ambienti, le nostre scorciatoie per comporre meglio e più correttamente.

Gli ultimi esempi sono piuttosto ricchi di spunti, ma non esauriscono certamente la questione della creazione delle proprie macro. Questa è un'arte difficile, ma non impossibile; bisogna avere la pazienza e l'umiltà di imparare senza fretta, sapendo però che la ricompensa sarà forte sotto l'aspetto della facilità del comporre.

Bisogna avere la pazienza di imparare dai libri e dai manuali esistenti, alcuni dei quali elencati nell'appendice A. Bisogna avere anche la pazienza di leggere il codice dei file scritti da altri, in primis quelli che definiscono il formato e le classi e i pacchetti standard. Alcune parti sono veramente difficili da capire senza l'aiuto di una documentazione, ma questa è generalmente disponibile nei file `.dtx`; questi sono file TeX documentati. Se si lancia LaTeX o pdfLaTeX su uno qualunque di questi file, si ottiene un documento nel quale il codice è inframezzato a spiegazioni; se queste siano utili dipende dalla competenza informatico/programmatoria del lettore, ma è certo che, superato un primo scoglio per familiarizzarsi con il linguaggio, la cosa diventa abbastanza semplice e fruttuosa.

Appendice A

Dove documentarsi

Invece di inserire un anonimo elenco bibliografico, qui si cerca di esporre una lista ragionata di testi utili e/o necessari per approfondire la conoscenza di LaTeX e dei suoi fratelli.

A.1 Documentazione sulla tipografia

Questa sezione avrebbe l'ambizione di fornire qualche idea sulla tipografia, le consuetudini tipografiche e il book design. Per ovvi motivi questa sezione è abbondantemente incompleta e la scelta dei riferimenti è frutto di un compromesso.

Una breve guida sulle consuetudini della tipografia italiana è contenuta nell'articolo:

Gustavo Cevolani, "Norme tipografiche", ArsTeXnica, (2006), n. 1.

Questo articolo ha senz'altro il pregio della concisione, una decina di pagine, ma è piuttosto completo e indica anche come le varie consuetudini tipografiche italiane possono essere rese con i comandi e gli ambienti LaTeX.

Benché abbia i suoi anni, il manuale:

Roberto Lesina, *Il nuovo manuale di stile*, ed. 2.0, Zanichelli, Bologna, 1994

raccoglie in modo ben strutturato un certo numero di consuetudini tipografiche e raccomandazioni di vario genere, che molti scrittori tecnico–scientifici considerano assai valide.

Anche la documentazione della classe `memoir`:

Peter Wilson, *The Memoir Class for Configurable Typesetting — User Guide*, The Herries Press, Normandy Park, WA, "Printed in the World", 2004, in `.../doc/latex/memoir/`

contiene, oltre al manuale d'uso per la classe stessa, non poche pagine di iniziazione alla tipografia e al book design. Anche se il testo è in inglese e si riferisce principalmente alla tipografia inglese, non mancano riferimenti alla tipografia europea continentale di varie nazioni e di vari secoli.

Il testo elettronico:

PETER FLYNN, *Formatting information — A beginner's introduction to typesetting with LaTeX*, `beginlatex-3.6.pdf`; questo file, eventualmente in versione successiva alla 3.6, può essere scaricato dagli archivi CTAN.

è sostanzialmente un guida all'uso di LaTeX, ma l'autore è professore di Tipografia all'University College Cork, Irlanda. Fu lui il Presidente e organizzatore della Conferenza internazionale di Cork, dove furono gettate le basi per l'encoding esteso T1. Data la professione dell'autore, il testo è particolarmente curato nella sua forma grafica e compositiva; è vero, ci sono delle convenzioni insolite; per esempio i richiami delle note seguono la punteggiatura, mentre nella maggior parte dei libri italiani i richiami di nota precedono la punteggiatura. Tuttavia il tipo di composizione è piuttosto curata, certamente può costituire un esempio da seguire.

La Imprimerie Nationale francese, l'equivalente del nostro Poligrafico dello Stato, ha pubblicato il libretto:

AUTORI VARI, *Lexique des règles typographiques en usage à l'Imprimerie Nationale*, Imprimerie Nationale, Parigi?, 2006, ISBN 2-7433-0482-0.

Si tratta di una specie di glossario dove sono elencate in orine alfabetico le varie parole che comportano certe consuetudini tipografiche e viene specificato come quelle consuetudini siano applicate presso questo autorevolissimo ente.

Tipica della tradizione tipografica francese è la spaziatura uniforme fra le parole indipendentemente da quali segni di interpunzione le separino; questo effetto con LaTeX si ottiene con la dichiarazione `\frenchspacing`. Vale la pena di segnalare che questo tipo di spaziatura è quello di default quando viene composta la bibliografia; in questo capoverso si è appunto specificata la spaziatura francese.

Vale la pena di osservare che le tradizioni tipografiche francesi sotto certi spetti sono simili a quelle italiane, e sotto certi altri sono molto divergenti; non è solo una questione di lingua, ma si tratta proprio di consuetudini, una volta, molti decenni fa, usate anche in Italia. Per esempio gli spazi prima di tutti i segni di interpunzione 'alti', dai due punti al punto esclamativo e interrogativo; in questo libretto, per altro composto con cura, a me pare assai sgradevole il fatto che lo spazio che precede questi segni sia piuttosto grande e suscettibile di essere ulteriormente allargato per giustificare le righe; forse la sgradevole sensazione dipende dalla mancanza di abitudine. Questa sensazione mi si ripresenta quando vedo i numeri romani usati come ordinali ma con la desinenza ad apice, come in 'Lille est le siège de la II^{e} région militaire'; in italiano sarebbe certamente scorretto mettere ad apice di un numero romano la desinenza dell'aggettivo ordinale.

Bisogna però ammettere che questo è uno dei pochi manuali di composizione che tratta anche gli aspetti che non vengono mai trattati dagli autori più famosi; per esempio la composizione della matematica, l'uso delle unità di misura, la scrittura della chimica, anche quella organica con le sue formule di struttura. Anche in matematica la tipografia francese si distingue con particolarità tutte sue, tuttavia le indicazioni fornite in questo libro sono certamente una buona guida anche per gli italiani.

Abbastanza documentato e facile anche per i neofiti è il libro:

MICHAEL MITCHEL, SUSAN WIGHTMAN, *Book Typography — A Designer's manual*, Libanus Press, Marlborough, Wiltshire UK, 2005.

Esso copre tutti gli aspetti della produzione di un libro, e dà indicazioni assai utili a chi vorrebbe fare o fa di professione il book designer, sia pure in un contesto britannico. In realtà fuori dal Regno Unito, cambiano i dettagli, ma i problemi da affrontare per la produzione di un libro sono praticamente gli stessi in ogni nazione. Un dilettante legge volentieri questo manuale, perché non è formato da una serie di prescrizioni, ma è un susseguirsi di descrizioni dei vari aspetti, inclusi quelli tipografici e compositivi, naturalmente, ma è molto istruttivo. Gli autori non fanno un mistero di avere usato un prodotto professionale e commerciale per la *mise en page*, come direbbero i francesi, ma si sforzano di dare indicazioni per non vincolare il lettore allo stesso programma di impaginazione. Molti dei problemi da risolvere a mano con gli impaginatori professionali, sono risolti automaticamente con LaTeX, ma ce ne sono altri che con LaTeX sarebbe molto difficile affrontare e risolvere. È utile domandarsi via via: "Questo con LaTeX come potrebbe essere fatto?"

Fra i grandi book designer di questo secolo compare anche Robert Bringhurst; egli ha scritto un piacevole libro

Robert Bringhurst, *The Elements of Tpypographic Style*, ver. 3.0, Hartley & Marks, Vancouver BC, Canada, 2004

dove egli descrive i suoi principi tipografici e di book and page design; è incredibile come riesca a legare le proporzioni delle note musicali delle scale diatonica e cromatica con quelle delle figure geometriche e con quelle da usarsi in tipografia. Nelle sue mani quei principi per la scelta delle proporzioni diventano davvero musica, per gli occhi, invece che per le orecchie, ma con quella sobrietà che consente alla persona di gusto di apprezzarne le finezze, senza però imporsi fino a distrarre il lettore. Leggendo questo libro e comprendendo appieno i principi tipografici di Bringhurst, si capisce che il book design di qualità non è un gioco da dilettanti.

Gli ultimi riferimenti sono testi in inglese; è un peccato che non esistano o siano difficili da trovare opere simili in italiano; d'altra parte è anche comprensibile che la pubblicazione di opere di questo genere in italiano per lettori italiani non sarebbe remunerativo, mentre lo è certamente se sono pubblicate in inglese per i lettori che usano l'inglese come prima o come seconda lingua.

Tra le opere italiane si può citare

Giorgio Fioravanti, *Il manuale del grafico — Guida alla progettazione grafica e all'impaginazione del prodotto editoriale*, Zanichelli Editore, Bologna, 1987.

Come dice il titolo, questo libro è più rivolto agli aspetti grafici, piuttosto che a quelli compositivi; penso che si tratti di una buona lettura, ma certamente non ha il respiro dell'opera di Bringhurst o la copertura di argomenti di Mitchel e Wightman. Tuttavia merita notare la composizione moderna a gabbia su tre colonne, dove spesso due delle colonne sono unite assieme, così che le pagine sono mosse e attraenti; in questa apparente disomogeneità la struttura grafica si adegua al contenuto, senza però attrarre l'attenzione su di sé distraendo il lettore dal contenuto.

Il libro veniva usato qualche anno fa come libro di testo all'Istituto Steiner di Torino proprio per insegnare la grafica editoriale agli allievi di grafica pubblicitaria.

A.2 Documentazione su LaTeX

La fonte iniziale di ogni sapere a proposito di LaTeX standard è il manuale di Leslie Lamport

Leslie Lamport, *A document preparation system — LaTeX — User's guide and reference manual*, Addison Wesley Publ. Co., Reading Mass., 2 ed., 1994.

Per leggere un testo introduttivo più breve del manuale di Lamport, si può ricorrere a

Tobia Öticker, *Una (mica tanto) breve introduzione a LaTeX 2ε Ovvero LaTeX 2ε in 93 minuti*, `.../guides/lshort/itlshort.pdf`, 2000, Traduzione di G. Agostini, G. Bilotta, F. Casadei Della Chiesa, O. de Bari, G. Delre, L. Ferrante, T. Pecorella, M. Rigido, R. Zanasi.

Invece, per andare su un testo più completo anche della Guida di Lamport ci si può riferire all'eccellente

Helmut Kopka, Patrick W. Daly, *Guide to LaTeX*, Addison Wesley Publ. Co., Reading Mass., 4 ed. 2004.

Ma il libro più completo (e pesante, non nel prezzo, ma nel numero di pagine), il libro dei libri su LaTeX, è certamente

Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley et al., *The LaTeX companion*, Addison Wesley Publ. Co., Reading Mass., 2 ed., 2004.

Si tratta di un libro di quasi 1100 pagine dove c'è tutto lo scibile concernente LaTeX 2ε e i più importanti pacchetti.

A.3 Documentazione sulla grafica

Sta per essere pubblicata anche la seconda edizione del volume

Frank Mittelbach, Michel Goossens et al., *The LaTeX graphics companion*, Addison Wesley Publ. Co., 2 ed., in fase di pubblicazione alla fine del 2006.

Tutti i libri della Addison Wesley sono acquistabili tramite il sito dell'associazione mondiale degli utenti di TeX con notevoli sconti; si suggerisce di esplorare il sito `http://www.tug.og/books/`.

A.4 Documentazione sui singoli pacchetti

La documentazione sui singoli pacchetti di estensione fa generalmente parte della distribuzione stessa del pacchetto; ogni volta che si scarica dalla rete un pacchetto e questo viene installato nella struttura standard di cartelle del sistema TeX, la documentazione va a finire in una cartella che si chiama come il pacchetto nel ramo dell'albero che comincia con `.../doc`; può poi finire nella ramo `latex` oppure `tex` oppure `generic` a seconda che l'uso del pacchetto sia specifico di LaTeX, di TeX, oppure sia polivalente.

Queste cartelle sono le prime a dover essere esaminate quando si cerca della documentazione specifica; non lo si raccomanda mai abbastanza, tanto che molti utenti si rivolgono in rete alla prima mailing list che capita per avere delucidazioni su cose che sono spiegate chiarissimamente sulla documentazione che arriva insieme al sistema TEX e che quindi è già installata nello stesso calcolatore dove l'utente si è trovato di fronte ad un problema. Per carità un po' di aiuto non lo si nega a nessuno, ma la 'netiquette' impone di chiedere aiuto prima di tutto alla mailing list competente e solo dopo aver fatto il possibile per documentarsi autonomamente.

Per i problemi relativi al sistema TEX è certamente competente il forum `comp.text.tex` in `groups.google.com`.

Anche il Gruppo degli utilizzatori Italiani di TEX ha un forum all'indirizzo `http://www.guit.ssusp.it/forum/` ed è meglio rivolgersi a questi gruppi, invece di scrivere alla mailing list dello shell editor che si sta usando, dimostrando quindi di non aver capito la distinzione fra lo shell editor e il sistema TEX.

Ma una cartella che bisogna avere sempre presente è `.../doc/latex/base/` perché quella cartella contiene alcune rapide guide dal titolo *xxx*`guide.dvi`, dove le tre lettere *xxx* indicano un po' stenograficamente di che cosa si tratti: *usr* riguarda il generico utente, *cls* riguarda le classi, eccetera; anche se sono abbreviazioni inglesi non è difficile decifrarle.

Un problema per gli utenti italiani è costituito dal fatto che quasi tutta la documentazione è scritta in inglese; capisco che non tutti sono tenuti a conoscere l'inglese come la propria lingua madre, tuttavia se si usa un PC o un laptop, se si naviga in rete, una conoscenza di base dell'inglese è sicuramente presente; questa conoscenza di base dell'inglese scritto dovrebbe essere sufficiente.

A.5 Documentazione su TEX

Se si arriva a scrivere dei file di classe o dei pacchetti di estensione viene voglia di conoscere meglio il linguaggio TEX primitivo; il libro dei libri è quello scritto direttamente dal padre del sistema TEX

DONALD E. KNUTH, *The TEXbook*, Addison Wesley Publ. Co., Reading Mass., 16 ed. 1986 (in realtà la 16ª edizione contiene correzioni e aggiunte datate 1996).

Questo libro è la fonte più autorevole sul sistema TEX; se si è in vena di spese esiste anche l'*opera omnia* di Knuth relativa alla tipografia digitale:

DONALD E. KNUTH, *Computers & Typesetting Millenium Edition*, Addison Wesley Publ. Co., Reading Mass., 2001.
Contiene i cinque volumi: Volume A, *The TEXbook*; Volume B, *TEX: The Program*; Volume C, *The METAFONTbook*; Volume D, *METAFONT: The Program*; Volume E, *Computer Modern Typefaces*.

Questa Millennium Edition in un elegante cofanetto cartonato contiene i tutti e cinque i libri del sistema TEX. Tutti e cinque i volumi sono incassati (copertina rigida) e, a seconda dell'uso che se ne fa, sono molto meglio che non procurarsi separatamente le varie parti, per giunta brossurate o con legature a spirale.

A.6 Documentazione sui simboli di LaTeX

LaTeX consente di inserire nei documenti composti una miriade di simboli che sono disponibili con certi pacchetti oppure con certi font; esiste un documento in rete, predisposto da Scott Pakin e tenuto costantemente aggiornato, che contiene tutti i simboli gestibili con LaTeX; si tratta di

Scott Pakin, *The Comprehensive LaTeX Symbol List*, file in
`http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf`

L'indirizzo in rete è sempre lo stesso dopo ogni aggiornamento.

A.7 Documentazione sulla composizione della matematica

In una lettera privata a un suo corrispondente Jean-Paul Serre ha scritto:

> Mi colpisce il fatto che la scrittura matematica sia simile alla scrittura di una lingua. Per essere capito devi seguire certe regole grammaticali. Tuttavia, nel nostro caso, nessuno si è preso la briga di scrivere una grammatica; noi l'afferriamo come un bimbo lo fa dai suoi genitori, per imitazione degli altri. Alcuni matematici hanno un buon orecchio; altri no (e alcuni preferiscono usare espressioni gergali come, per esempio, "iff"). Così è la vita.[1]

Tuttavia non è proprio completamente esatta questa affermazione; alcune regole esistono, almeno per certi stili di scrittura matematica.

Le norme ISO per la composizione della matematica si trovano tradotte in italiano nel documento pubblicato dall'Ente di Unificazione Italiano dal titolo

Segni e simboli matematici per le scienze fisiche e tecniche, CNR–UNI 10002, Milano, Unificazione Italiana, 1963

Vale la pena di consultare anche le indicazioni fornite dalla Unione Internazionale di Fisica Pura e Applicata:

"Symbols, units and nomenclature in physics" in *CRC Handbook of chemistry and physics*, a cura di Weast R.C., Astle M.J., Beyer W.H., Boca Raton, Florida, CRC Press Inc., 65 ed., p. F259–F293, 1984.

L'American Mathematical Society ha anche pubblicato un libretto su questo argomento per dare adeguate istruzioni agli autori che le inviano lavori matematici da pubblicare:

Ellen Swanson, Arlene O'Sean, Antoinette Schleyer, *Mathematics into type*, American Mathematical Society, Providence RI, 1999 (Edizione riveduta).

[1] It strikes me that mathematical writing is similar to using a language. To be understood you have to follow some grammatical rules. However, in our case, nobody has taken the trouble of writing down the grammar; we get it as a baby does from its parents, by imitation of others. Some mathematicians have a good ear; some not (and some prefer the slangy expressions such as "iff"). That's life.

Come appare chiaro dal titolo, i primi riferimenti si riferiscono esplicitamente alla matematica scritta dai fisici e dai tecnologi, mentre l'ultimo si riferisce alla matematica dei matematici; come è stato giustamente osservato, sarebbe opportuno che fisici, matematici e ingegneri scrivessero la matematica nello stesso modo. Non c'è dubbio! Ma non conoscendo se il lettore sia un matematico, o un fisico, o un ingegnere, o un economista, o un giurista, eccetera, allora ...

In ogni caso si rinvia all'appendice C una lista di simboli e di notazioni tratta dalle norme o dalle pubblicazioni suddette.

Appendice B

La composizione di testi filologici

La composizione di testi filologici è forse un argomento troppo specializzato per un guida introduttiva come questa.

Tuttavia dopo aver dedicato due capitoli alla matematica, sembra uno sgarbo evitare la filologia che è una scienza molto esatta e che si avvale di una notazione bidimensionale come la matematica.

La pubblicazione di scritti filologici include le edizioni critiche e include scritti in prosa o in versi in lingue scritte non necessariamente in caratteri latini.

Essi richiedono diversi apparati di note che spesso compaiono nella stessa pagina ma devono venire distinti in modo inequivocabile gli uni dagli altri; chiaramente non basta solo un filetto orizzontale per distinguere questi apparati di note.

Serve anche una numerazione delle singole righe sia per i testi in prosa, sia per quelli in poesia; generalmente i numeri sono nel margine esterno e sono espliciti come multipli di 5; le note, invece che legate a richiami come numerini a esponente, possono essere legate al numero di riga, per cui nel richiamo bisogna ripetere una breve sequenza di testo a cui la nota si riferisce.

Chiaramente LaTeX non è in grado da solo di svolgere questi compiti e bisogna ricorrere a pacchetti di estensione.

Il più noto pacchetto di estensione si chiama **ledmac** ed è in grado di comporre ogni aspetto delle edizioni critiche; esso è documentato in `.../doc/latex/ledmac/ledmac.dvi`.

L'autore Peter Wilson ha cercato con **ledmac** di implementare le capacità sviluppate dai pacchetti di macro per plain TeX **edmac**, **tabmac** e **edstanza** aggiungendovi del suo.

Esso è particolarmente indicato per le edizioni critiche e il suo forte sono le numerazioni marginali e gli apparati di note sia a piè di pagina sia come note finali a fine capitolo o a fine documento. È possibile legare le note finali al numero di pagina e di riga, invece che a richiami numerici ad esponente. Anche per quel che riguarda l'indice analitico è possibile fare riferimento anche al numero di riga invece che solamente alla pagina.

Altri pacchetti per edizioni critiche sono **endnotes** specializzato nelle note finali, e **poemscol** destinato alle forme poetiche, ai testi in versi.

Quando si va nella filologia vera e propria, le cose si possono fare molto dure, anche perché i filologi delle varie discipline hanno notazioni diverse per rendere gli stessi concetti.

Chi scrive ha collaborato con un filologo di greco classico e con un filologo di lingua copta altomedievale; ovviamente i due filologi non erano d'accordo su come esprimere i concetti comuni alle due lingue; questa non è una critica, ma una osservazione; anch'io che scrivo usando molta matematica ho dovuto arrangiarmi con i simboli disponibili perché certi concetti che dovevo esprimere, malgrado le norme ISO, non avevano un simbolo specifico, così alla fine ho dovuto usare simboli che in altre branche della matematica hanno significati diversi. Succede in ogni scienza e anche all'interno della stessa scienza.

Per quanto concerne il greco classico esiste il pacchetto **teubner** scaricabile dalla rete e abbastanza ben documentato mediante il file `.../doc/latex/teubner/teubner-doc.pdf`.

Il pacchetto consente di accentare ogni lettera con qualsiasi segno diacritico, persino con accenti impilati, collocandoli correttamente tenendo conto dell'inclinazione dei caratteri. Esso può inserire segni speciali sopra o sotto singole lettere o gruppi di lettere; può racchiudere gruppi di lettere con delimitatori speciali per indicare la lectio incerta, o altre simili indicazioni per interpolazioni o omissioni; può numerare i versi di poemi anche con due serie di numerazioni contemporaneamente; può disegnare le metre dei vari versi con una notevole ricchezza di segni che consentono anche di specificare, per esempio, se una sillaba ancipite è più sovente lunga invece che breve, eccetera.

Per quanto concerne il copto, oltre ai font necessari per scrivere con i caratteri altomedievali, ci sono una certa varietà di simboli e segni che permettono di annotare il testo antico; si può scaricare il tutto da `CTAN/language/coptic/coptic.zip`. Il file `README` contenuto nel file compresso spiega come decomprimere il pacchetto e dove sistemare ogni componente del pacchetto.

Certamente esistono altri pacchetti in rete destinati a risolvere problemi specifici di composizione filologica; tuttavia ogni lavoro di filologia richiede la predisposizione di comandi particolari per problemi particolari; lo scrivente è al corrente che da alcuni anni a Pisa si sta costruendo l'edizione critica delle opere del matematico Maurolico; sono già disponibili diverse parti, ma l'opera non è finita; si capisce bene che, nonostante il Maurolico scrivesse in latino, egli scriveva di matematica e di geometria quando le convenzioni a cui siamo oggi abituati non esistevano ancora, non parliamo delle norme ISO! I curatori di questa edizione critica stanno lavorando con passione ma devono inventarsi soluzioni nuove per problemi di composizione che finora non sono mai stati affrontati, meno che mai risolti.

In Italia, oltre che a Pisa, si è al corrente che anche alla Pontificia Università di Santa Croce a Roma c'è un gruppo di utenti che si occupa di filologia e di edizioni critiche. Essi hanno anche predisposto un corso e un CD-ROM per gli utenti del corso.

Tuttavia, come si diceva, l'argomento è estremamente specializzato e forse anche questa appendice è fuori luogo in un testo introduttivo.

Appendice C

Simbologia matematica e fisica

C.1 Unità di misura del Sistema Internazionale

Le unità fondamentali del Sistema Internazionale sono raccolte nella tabella C.1; secondo le norme internazionali gli angoli piani e solidi sono considerati "quantità derivate adimensionate"; infatti le norme affermano che "le unità *radiante* e *steradiante* devono essere considerate come unità derivate adimensionate che possono essere usate od omesse nelle espressioni delle unità derivate". È per questo che più avanti per alcune grandezze fisiche saranno indicate *fra parentesi* le unità di misura contenenti anche i radianti o gli steradianti in quei casi in cui il loro uso consente di distinguere specie fisiche diverse e apparentemente equidimensionate.

Grandezza fisica	Unità	Simbolo
lunghezza	metro	m
massa	kilogrammo	kg
tempo	secondo	s
corrente elettrica	ampere	A
temperatura termodinamica	kelvin	K
quantità di sostanza	mole	mol
intensità luminosa	candela	cd

Tabella C.1: Unità fondamentali

Tutte queste unità, nonché quelle delle tabelle successive, possono essere precedute dai prefissi decimali raccolti nella tabella C.2. Si ricorda che i prefissi vanno usati isolatamente (in passato non era infrequente osservare il prefisso *millimicro* al posto del prefisso corretto *nano*). Quando l'unità di misura con prefisso è elevata ad un esponente, questo si intende applicato all'unità completa di prefisso: $3\,\mathrm{cm}^3$ indica un volume di $3\,(10^{-2}\mathrm{m})^3 = 3\times10^{-6}\,\mathrm{m}^3$ e non un volume di $3\times10^{-2}\,\mathrm{m}^3$.

Quando si parla di byte (simbolo B) si usano i prefissi binari; questi sono riportati nella tabella C.3; essi sono legali dal 1998 e dovrebbero venire sempre

usati quando si parla di quantità "informatiche", come le capienze dei dischi, o le dimensioni di certe memorie, e simili. Si noti che per l'unità kibi (kilo binario) il simbolo comincia con una K maiuscola; non è un errore: tutti i prefissi binari cominciano con lettere maiuscole incluso quello che ricorda il prefisso kilo; lo si ricorda solamente, ma Ki sta per 1024, mentre k sta per 1000; sono evidentemente due cose diverse.

Prefisso	Valore	Simbolo	Prefisso	Valore	Simbolo
yotta	10^{24}	Y	deci	10^{-1}	d
zetta	10^{21}	Z	centi	10^{-2}	c
exa	10^{18}	E	milli	10^{-3}	m
peta	10^{15}	P	micro	10^{-6}	µ
tera	10^{12}	T	nano	10^{-9}	n
giga	10^{9}	G	pico	10^{-12}	p
mega	10^{6}	M	femto	10^{-15}	f
kilo	10^{3}	k	atto	10^{-18}	a
etto	10^{2}	h	zepto	10^{-21}	z
deca	10^{1}	da	yocto	10^{-24}	y

Tabella C.2: Prefissi decimali

Prefisso	Valore	Simbolo
kibi	2^{10}	Ki
mibi	2^{20}	Mi
gibi	2^{30}	Gi
tebi	2^{40}	Ti
pebi	2^{50}	Pi
exbi	2^{60}	Ei

Tabella C.3: Prefissi binari

Per quanto riguarda le cosiddette unità logaritmiche, sono codificate quelle della tabella C.4; si ricorda che i nomi che vengono dati a queste unità servono solo a ricordare quale base è stata usata per il calcolo del logaritmo. Si richiama l'attenzione anche sulla corretta scrittura dei simboli dB e Np, che invece si vedono così spesso scritti in modo errato. Per quanto riguarda le unità di attenuazione e di guadagno si usano i logaritmi decimali per i decibel, o neperiani per i neper, ed in più si hanno definizioni diverse a seconda che il rapporto di cui si calcola il logaritmo sia eseguito fra grandezze di potenza o energia, oppure fra grandezze di campo:

$$\alpha = 10 \log_{10} \frac{P_1}{P_2} \qquad \text{oppure} \qquad \alpha = \frac{1}{2} \log_{\mathrm{e}} \frac{P_1}{P_2}$$

dove P_1 e P_2 sono potenze, oppure

$$\alpha = 20 \log_{10} \frac{V_1}{V_2} \qquad \text{oppure} \qquad \alpha = \log_{\mathrm{e}} \frac{V_1}{V_2}$$

dove V_1 e V_2 sono tensioni.

Grandezza	Unità	Simbolo
attenuazione, guadagno	decibel	dB
attenuazione, guadagno	neper	Np
intervallo di frequenza	ottava	ott
intervallo di frequenza	decade	dec

Tabella C.4: Unità logaritmiche

Per gli intervalli di frequenza si usano i logaritmi binari per le ottave, o decimali per le decadi

$$I = \log_2 \frac{f_2}{f_1} \qquad \text{oppure} \qquad I = \log_{10} \frac{f_2}{f_1}$$

È stato necessario introdurre molte altre unità per le grandezze fisiche derivate, al fine di evitare di dover usare lunghi elenchi di unità fondamentali elevate a potenze insolite, che sarebbe fra l'altro troppo complicato ricordare; queste unità derivate sono elencate nella tabella C.5.

Grandezza fisica	Unità	Simbolo
angolo piano	radiante	rad
angolo solido	steradiante	sr
frequenza	hertz	Hz
forza	newton	N
pressione	pascal	Pa
lavoro, energia	joule	J
potenza	watt	W
carica elettrica	coulomb	C
tensione elettrica	volt	V
capacità elettrica	farad	F
resistenza elettrica	ohm	Ω
conduttanza elettrica	siemens	S
flusso di induzione magnetica	weber	Wb
induzione magnetica	tesla	T
induttanza	henry	H
flusso luminoso	lumen	lm
illuminamento	lux	lx
attività di un radionuclide	becquerel	Bq
dose assorbita	gray	Gy
equivalente di dose	sievert	Sv
attività catalitica	katal	kat

Tabella C.5: Unità derivate

Le necessità della vita civile e le esigenze del commercio hanno obbligato ad ammettere molte unità aggiuntive, che spesso sono dei duplicati in scala diversa delle unità fondamentali o derivate; per esempio il *carato metrico* è un'altra unità di massa che si affianca al *kilogrammo* (ed ai suoi sottomultipli) e che sarebbe del tutto superflua, ma è stata conservata per rispettare una tradizione

Grandezza fisica	Unità	Simbolo	Equivalenza
angolo piano	grado sessagesimale	°	$1° = \pi/180\,\mathrm{rad}$
angolo piano	minuto sessagesimale	′	$1′ = \pi/10\,800\,\mathrm{rad}$
angolo piano	secondo sessagesimale	″	$1″ = \pi/648\,000\,\mathrm{rad}$
angolo piano	gon o grado centesimale	gon	$1\,\mathrm{gon} = \pi/200\,\mathrm{rad}$
angolo piano	giro	giro	$1\,\mathrm{giro} = 2\pi\,\mathrm{rad}$
area	ara	a	$1\,\mathrm{a} = 100\,\mathrm{m}^2$
area	ettaro	ha	$1\,\mathrm{ha} = 10\,000\,\mathrm{m}^2$
volume	litro	l, L, ℓ	$1\,\ell = 1\,\mathrm{dm}^3$
tempo	minuto	min	$1\,\mathrm{min} = 60\,\mathrm{s}$
tempo	ora	h	$1\,\mathrm{h} = 3600\,\mathrm{s}$
tempo	giorno	d	$1\,\mathrm{d} = 86\,400\,\mathrm{s}$
massa	tonnellata	t	$1\,\mathrm{t} = 1000\,\mathrm{kg}$
massa	carato metrico	carato metrico	$1\,\text{carato metrico} = 200\,\mathrm{mg}$
massa	unità di massa atomica	u	$1\,\mathrm{u} = 1{,}660\,57 \cdot 10^{-27}\,\mathrm{kg}$
massa lineica	tex	tex	$1\,\mathrm{tex} = 1\,\mathrm{mg/m}$
pressione	bar	bar	$1\,\mathrm{bar} = 10^5\,\mathrm{Pa}$
lavoro, energia	elettronvolt	eV	$1\,\mathrm{eV} = 1{,}602\,19 \cdot 10^{-19}\,\mathrm{J}$
lavoro, energia	kilowattora	kW h	$1\,\mathrm{kW\,h} = 3{,}6\,\mathrm{MJ}$
carica elettrica	amperora	A h	$1\,\mathrm{A\,h} = 3600\,\mathrm{C}$
temperatura Celsius	grado Celsius	°C	$1\,°\mathrm{C} = 1\,\mathrm{K}$ ma differisce lo zero della scala: $t = T - 273{,}15\,\mathrm{K}$

Tabella C.6: Unità di misura legalmente ammesse

in un settore merceologico dove non è possibile nessuna interferenza con l'unità di massa ordinaria.

Un cenno particolare merita il *litro* perché sono leciti ben tre simboli per questa unità: l, L e ℓ; il terzo è il simbolo che l'Unione Europea ha prescritto per tutte le affermazioni di carattere merceologico e costituisce una scelta quanto mai opportuna, perché evita ogni possibile confusione della l minuscola con la cifra 1, e quella della L maiuscola con la cifra 4.

Le unità ammesse sono riportate nella tabella C.6.

Infine sono ancora *tollerate* un certo numero di altre unità *in via di estinzione*; quelle che si sono già estinte (come l'atmosfera, il quintale, il millimetro di mercurio — ammesso solo in campo medico —, il poise, eccetera) non sono nemmeno elencate proprio per evitare che possa venire la tentazione di usarle ancora. Le convenzioni internazionali, a cui l'Italia aderisce, faranno sparire in un prossimo futuro anche queste unità tollerate, che sono elencate nella tabella C.7.

Grandezza fisica	Unità	Simbolo	Equivalenza
lunghezza	miglio marino	miglio marino	$1\,\text{miglio marino} = 1852\,\text{m}$
lunghezza	ångström	Å	$1\text{Å} = 10^{-10}\,\text{m}$
area	barn	barn	$1\,\text{barn} = 10^{-28}\,\text{m}^2$
velocità	nodo	nodo	$1\,\text{nodo} = (4{,}63/9)\,\text{m/s}$
			$1\,\text{nodo} = 1\,\text{miglio marino/h}$
accelerazione	gal	Gal	$1\,\text{Gal} = 1\,\text{cm/s}^2$

Tabella C.7: Unità di misura tollerate

Nelle tabelle C.1 – C.7 si notano delle assenze vistose, oltre a quelle già segnalate; in particolare mancano tutte le unità CGS, dagli erg alle dine, dai gauss agli örsted, tanto per citare quelle più comuni; si notano inoltre le assenze dei simboli cc, mc, mmc, mq, che sono scorrettamente tanto comuni in alcune scienze; al loro posto vanno usati i simboli corretti cm^3, m^3, mm^3, m^2. Una volta, ai tempi della dattilografia, era piuttosto laborioso inserire gli esponenti, oggi con i sistemi di videoscrittura, in particolare di composizione tipografica, come LaTeX, gli esponenti non sono più un problema, quindi quelle abbreviazioni errate, tollerate per necessità di cose fino a trenta anni fa, oggi non sono più accettabili.

C.2 Simboli matematici nelle scienze

In questo paragrafo sono raccolti i simboli matematici più comuni che si impiegano nelle scienze e nella fisica; essi sono ispirati tra l'altro alle norme CNR UNI 10002 e alle norme CEI 24-1, ma, quando queste norme erano in conflitto, è stata operata una scelta arbitraria se usare il simbolo proposto da una norma piuttosto che quello indicato dall'altra, oppure se indicarli entrambi. È stato seguito questo criterio anche per la scelta dei simboli che si ritengono meno frequenti e che non sono stati inseriti nella tabella C.8.

Come al solito l'elenco non è e non può essere completo, ma può servire da guida o modello per preparare un analogo elenco qualora si facesse uso di una matematica piuttosto elaborata.

Nella tabella C.8 a e b sono due numeri reali qualsiasi, i, j, k, n sono numeri interi, z, s sono variabili o numeri complessi, x, y, (talvolta anche z), e t sono variabili reali, D è un dominio, A, B, C, P sono punti del piano o dello spazio. La colonna intestata **Simbolo** contiene il segno grafico del simbolo, oppure un'espressione che ne fa uso.

C.3 Nomenclatura

È praticamente impossibile fare un elenco di tutti nomi delle grandezze che vengono usate in ogni scienza, dalla fisica alla medicina, dall'elettronica alla geologia. Si ritiene però cosa utile riprendere l'elenco del prospetto IV della norma CNR–UNI 10003, ampliandolo un poco e aggiungendovi il simbolo (o una scelta di simboli) che sono comunemente accettati in ogni scritto scientifico, senza che sorga la necessità di compilare un elenco delle grandezze e dei simboli usati.

Nel compilare la tabella C.9 si è tratta ispirazione dalle norme CNR–UNI, dalle norme CEI, dal fascicolo CEI di nomenclatura nucleare, dal documento sulla nomenclatura pubblicato dalla Società Internazionale di Fisica, senza inventare nulla, ma operando solo delle scelte fra le grandezze o i simboli che sono stati inclusi o esclusi nella tabella.

Fra parentesi, nella colonna delle unità di misura, vi sono delle indicazioni ulteriori che comprendono anche i radianti o altre unità come i neper o i cicli, quando è parso che l'introduzione di queste unità accessorie rendesse più chiara la differenza fra grandezze di specie diversa ma apparentemente equidimensionate.

La tabella C.9, come detto sopra, è certamente incompleta, ma rappresenta comunque un modello da imitare qualora fosse necessario fare un elenco delle grandezze e dei simboli usati nel documento.

Tabella C.8: Simboli matematici

Simbolo	**Significato**	**Note**
,	virgola decimale	Non usare il punto per separare la parte intera dalla parte decimale. Non usare nemmeno altri separatori tra i gruppi di tre cifre prima e dopo la virgola. Il punto decimale si può usare solo scrivendo in inglese, ma i separatori fra gruppi di tre cifre devono essere assenti lo stesso. In entrambi i casi il separatore può essere costituito da uno spazio fine non separabile
∞	infinito	
π	$\pi = 3{,}141\,592\ldots$	Se il font lo consente, sarebbe desiderabile stampare la costante matematica π con un carattere non inclinato
e	$\mathrm{e} = 2{,}718\,281\ldots$	Va scritta in tondo, se non altro per distinguerla dalla carica elementare e; ma, come base degli esponenziali neperiani, sarebbe preferibile trattarla sempre come un operatore.
γ	$\gamma = 0{,}577\,215\ldots$	Vale la stessa osservazione fatta per π
i , j	unità immaginaria, operatore di rotazione	Va scritta in tondo come gli altri operatori
...	omissione	Si usa sia nel significato di *elementi omessi* sia in quello di *eccetera*
x, y, z	coordinate cartesiane	x: larghezza, y: profondità, z: altezza
ϱ, φ, z	coordinate cilindriche	ϱ: distanza dall'asse, φ: longitudine, z: altezza
$\varrho, \varphi, \vartheta$	coordinate sferiche	ϱ: distanza radiale, φ: longitudine, ϑ: colatitudine

continua

Continua tabella C.8

Simbolo	**Significato**	**Note**
$a = b$	uguale	
$a \neq b$	diverso	
$a \equiv b$	identico	
e $\approx 2{,}718$	uguale a circa	
$a \sim b$	proporzionale	Si può usare anche $a \propto b$
$a \leftrightarrow b$	equivalente	
$a > b$	maggiore	
$a < b$	minore	
$a \geq b$	maggiore o uguale	
$a \leq b$	minore o uguale	
$a \gg b$	molto maggiore	
$a \ll b$	molto minore	
$a \rightarrow b$	tendente	
$a \simeq b$	asintoticamente uguale	
$a \triangleq b$	corrispondente	Si usa nelle indicazioni di scala dei diagrammi: per esempio $1\,\mathrm{cm} \triangleq 10\,\mathrm{V}$
$a \div b$	intervallo	Si usa nel senso di "da a a b"
$a + b$	somma	
$a - b$	sottrazione	
$a\,b\,, a \cdot b$	moltiplicazione	Non usare altri simboli quando gli operandi sono indicati mediante lettere
$1{,}5 \times 2{,}3$	moltiplicazione	Non usare altri simboli quando gli operandi sono entrambi numerici

continua

Continua tabella C.8

Simbolo	**Significato**	**Note**
$\begin{cases} 1{,}5\,a \\ 1{,}5 \cdot a \end{cases}$	moltiplicazione	Gli operandi numerici precedono sempre quelli letterali
$a/b\,,\,\dfrac{a}{b}$	divisione	Le due simbologie possono essere mescolate; usare le parentesi per isolare le singole operazioni ed evitare ambiguità; per esempio $\dfrac{(a/b)+1}{(a/b)+(b/a)}$
$a \bmod b$	modulo	resto della divisione a/b con quoziente intero; è sempre $0 \le (a \bmod b)/b < 1$
a^b	elevazione a potenza	
$\sqrt[b]{a}$	estrazione di radice	Non usare né $\sqrt[b]{a}$ né $\sqrt[b]{(a)}$; se $b = 2$, b viene omesso
$\lvert a \rvert$	valore assoluto	
$\displaystyle\sum_{i=1}^{n} a_i$	somma	
$\displaystyle\prod_{i=1}^{n} a_i$	prodotto	
$n!$	fattoriale	
$\dbinom{n}{m}$	coefficiente binomiale	$\dfrac{n(n-1)\cdots(n-m+1)}{1 \times 2 \times \cdots m}$
$f(x)$	funzione	
$\log_b x$	logaritmo in base b	
$\log x$	logaritmo decimale	
$\ln x\,,\,\log_{\mathrm{e}} x$	logaritmo neperiano	

continua

Continua tabella C.8

Simbolo	**Significato**	**Note**
$\mathrm{lb}\,x$, $\log_2 x$	logaritmo binario	
e^x , $\exp x$	esponenziale	In questa e nelle funzioni successive scritte in caratteri tondi l'argomento non necessita di parentesi quando è composto da un solo elemento letterale o numerico
$\sin x$	seno	
$\cos x$	coseno	
$\tan x$	tangente	
$\cot x$	cotangente	
$\sinh x$	seno iperbolico	
$\cosh x$	coseno iperbolico	
$\tanh x$	tangente iperbolica	
$\coth x$	cotangente iperbolica	
$\arcsin x$	arcoseno	
$\arccos x$	arcocoseno	
$\arctan x$	arcotangente	
$\operatorname{arccot} x$	arcocotangente	
$\operatorname{arsinh} x$	arcoseno iperbolico	
$\operatorname{arcosh} x$	arcocoseno iperbolico	
$\operatorname{artanh} x$	arcotangente iperbolica	
$\operatorname{arcoth} x$	arcocotangente iperbolica	
$K(k)$	integrale ellittico completo di prima specie	$K(k) = \int_0^{\pi/2} \frac{\mathrm{d}\vartheta}{\sqrt{1 - k^2 \sin^2 \vartheta}}$
$F(\varphi, k)$	integrale ellittico incompleto di prima specie	$F(\varphi, k) = \int_0^{\varphi} \frac{\mathrm{d}\vartheta}{\sqrt{1 - k^2 \sin^2 \vartheta}}$

continua

Continua tabella C.8

Simbolo	**Significato**	**Note**
$E(\varphi, k)$	integrale ellittico incompleto di seconda specie	$E(\varphi,k) = \int_0^{\varphi} \sqrt{1-k^2\sin^2\vartheta}\,\mathrm{d}\vartheta$
$\Pi(n;\varphi,k)$	integrale ellittico incompleto di terza specie	$\Pi(n;\varphi,k) = \int_0^{\varphi} \frac{\mathrm{d}\vartheta}{(1-n\sin^2\vartheta)\sqrt{1-k^2\sin^2\vartheta}}$
φ	amplitudine	L'amplitudine è legata all'integrale ellittico incompleto di prima specie dalla relazione $x = F(\varphi,k)$
$\mathrm{sn}\,(x,k)$	seno ellittico	$\mathrm{sn}\,(x,k) = \sin\varphi$
$\mathrm{cn}\,(x,k)$	coseno ellittico	$\mathrm{cn}\,(x,k) = \cos\varphi$
$\begin{cases}\mathrm{dn}\,(x,k)\\ \Delta(\varphi)\end{cases}$	delta amplitudine	$\Delta(\varphi) = \sqrt{1-k^2\sin^2\varphi}$
$o(x)$	ordine di infinito o infinitesimo	Se $y = o(x)$ allora $\lim y/x = 0$
$O(x)$	ordine di infinito o infinitesimo	Se $y = O(x)$ allora $\lvert\lim y/x\rvert < \infty$
$\Gamma(z)$	funzione gamma	$\Gamma(z) = \int_0^{\infty} t^{z-1}\,\mathrm{e}^{-t}\,\mathrm{d}t$
$(a)_n$	simbolo di Pochhammer	$(a)_n = \frac{\Gamma(a+n)}{\Gamma(a)}$
$\mathrm{erf}\,(z)$	funzione d'errore	$\mathrm{erf}\,(z) = \frac{2}{\sqrt{\pi}}\int_0^{z} \mathrm{e}^{-t^2}\,\mathrm{d}t$
$\mathrm{erfc}\,(z)$	funzione complementare d'errore	$\mathrm{erfc}\,(z) = 1 - \mathrm{erf}\,(z)$
$\mathrm{C}(z)$	integrale di Fresnel	$\mathrm{C}(z) = \int_0^{z} \cos(\pi t^2/2)\,\mathrm{d}t$
$\mathrm{S}(z)$	integrale di Fresnel	$\mathrm{S}(z) = \int_0^{z} \sin(\pi t^2/2)\,\mathrm{d}t$
$\mathrm{Si}(z)$	seno integrale	$\mathrm{Si}(z) = \int_0^{z} \frac{\sin t}{t}\,\mathrm{d}t$
$\mathrm{Ci}(z)$	coseno integrale	$\mathrm{Ci}(z) = \gamma + \ln z + \int_0^{z} \frac{\cos t - 1}{t}\,\mathrm{d}t$

continua

Continua tabella C.8

Simbolo	**Significato**	**Note**
$E_1(z)$	esponenziale integrale	$E_1(z) = \int_z^\infty \frac{\mathrm{e}^{-t}}{t}\,\mathrm{d}t$
$\mathrm{Ei}(x)$	esponenziale integrale	$\mathrm{Ei}(x) = \int_{-\infty}^x \frac{\mathrm{e}^{-t}}{t}\,\mathrm{d}t$
$\mathrm{li}(x)$	logaritmo integrale	$\mathrm{li}(x) = \int_0^x \frac{\mathrm{d}t}{\ln t} = \mathrm{Ei}(\ln x)$
$\zeta(s)$	funzione Zeta di Riemann	$\zeta(s) = \sum_{k=1}^{\infty} k^{-s}$
$\delta(t)$	distribuzione di Dirac	
$\mathrm{u}(t)$	gradino unitario	$\mathrm{u}(t) = \begin{cases} 0 & \text{per } t < 0 \\ 1/2 & \text{per } t = 0 \\ 1 & \text{per } t > 0 \end{cases}$
δ_{ij}	simbolo di Kronecker	$\delta_{ij} = \begin{cases} 0 & \text{per } i \neq j \\ 1 & \text{per } i = j \end{cases}$
$J_\nu(z)$	funzione di Bessel di prima specie	
$Y_\nu(z)$	funzione di Bessel di seconda specie	
$H_\nu^{(1)}(z)$	funzione di Hankel di prima specie	$H_\nu^{(1)}(z) = J_\nu(z) + \mathrm{i}\,Y_\nu(z)$
$H_\nu^{(2)}(z)$	funzione di Hankel di seconda specie	$H_\nu^{(2)}(z) = J_\nu(z) - \mathrm{i}\,Y_\nu(z)$
$I_\nu(z)$	funzione di Bessel modificata di prima specie	
$K_\nu(z)$	funzione di Bessel modificata di seconda specie	
$\mathrm{ber}_\nu(x)$	prima funzione di Kelvin di prima specie	$\mathrm{ber}_\nu(x) = \mathbf{Re}\left[J_\nu\left(x\,\mathrm{e}^{3\pi\,\mathrm{i}\,/4}\right)\right]$
$\mathrm{bei}_\nu(x)$	seconda funzione di Kelvin di prima specie	$\mathrm{bei}_\nu(x) = \mathbf{Im}\left[J_\nu\left(x\,\mathrm{e}^{3\pi\,\mathrm{i}\,/4}\right)\right]$
$\ker_\nu(x)$	prima funzione di Kelvin di seconda specie	$\ker_\nu(x) = \mathbf{Re}\left[K_\nu\left(x\,\mathrm{e}^{\pi\,\mathrm{i}\,/4}\right)\right]$
$\mathrm{kei}_\nu(x)$	seconda funzione di Kelvin di seconda specie	$\mathrm{kei}_\nu(x) = \mathbf{Im}\left[K_\nu\left(x\,\mathrm{e}^{\pi\,\mathrm{i}\,/4}\right)\right]$

continua

Continua tabella C.8

Simbolo	Significato	Note
$M(a,b,z)$	funzione ipergeometrica confluente	Funzione di Kummer di prima specie
$U(a,b,z)$	funzione ipergeometrica confluente	Funzione di Kummer di seconda specie
$F(a,b;c;z)$	funzione ipergeometrica	L'espressione generale è $F(a,b;c;z) = \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)} \cdot \int_0^1 t^{b-1}(1-t)^{c-b-1}(1-tz)^{-a}\,\mathrm{d}t$
$P_n(z)$	polinomio di Legendre	Detto anche funzione sferica
$T_n(z)$	polinomio di Chebyshev di prima specie	
$U_n(z)$	polinomio di Chebyshev di seconda specie	
$C_n^{(\alpha)}(z)$	polinomio di Gegenbauer	Detto anche polinomio ultrasferico
$P_n^{(\alpha,\,\beta)}(z)$	polinomio di Jacobi	L'intervallo di ortogonalità è $-1 \div +1$
$G(p,q,z)$	polinomio di Jacobi	L'intervallo di ortogonalità è $0 \div +1$
$H_n(z)$	polinomio di Hermite	
$L_n(z)$	polinomio di Laguerre	
$L_n^{(\alpha)}(z)$	polinomio di Laguerre generalizzato	
$B_n(z)$	polinomio di Bernoulli	
$E_n(z)$	polinomio di Eulero	
$P_\nu^\mu(z)$	funzione ultrasferica di prima specie	Quando $\mu = 0$ si omette di scriverne il valore, perché la funzione coincide con il polinomio di Legendre
$Q_\nu^\mu(z)$	funzione ultrasferica di seconda specie	
$\lim_{x\to a} f(x)$	limite	
Δx	incremento finito	
δx	incremento virtuale	

continua

Continua tabella C.8

Simbolo	Significato	Note
$\mathrm{d}x$	differenziale	Come gli altri operatori va scritto in tondo, ma non richiede lo spazio a destra.
$f(x)\Big\vert_a^b$	incremento	Cioè $f(b) - f(a)$
$\dfrac{\mathrm{d}y}{\mathrm{d}x}$	derivata	
$\dfrac{\partial y}{\partial x}$	derivata parziale	
$\dfrac{\mathrm{d}^n y}{\mathrm{d}x^n}$	derivata n-esima	
$\dfrac{\partial^n y}{\partial x^n}$	derivata parziale n-esima	L'ordine di derivazione nelle derivate parziali miste, quando non sia indifferente, è il seguente $$\frac{\partial^2 z}{\partial x \partial y} = \frac{\partial}{\partial x}\left(\frac{\partial z}{\partial y}\right)$$
$\int f(x)\,\mathrm{d}x$	integrale indefinito	
$\int_a^b f(x)\,\mathrm{d}x$	integrale definito	
$\int_D f(\mathsf{P})\,\mathrm{d}D$	integrale esteso ad un dominio	Il particolare dominio D va specificato. Il punto P appartiene al dominio
$\fint_a^b f(z)\,\mathrm{d}z$	integrale principale di Cauchy	La funzione $f(z)$ è discontinua lungo la linea che congiunge a e b, e l'integrale viene calcolato come limite simmetrico a cavallo della discontinuità
$\oint_\ell f(z)\,\mathrm{d}z$	circuitazione di f lungo la linea chiusa ℓ	
$\mathcal{L}[f(t)]$	trasformata di Laplace	$F(s) = \mathcal{L}[f(t)]$
$\mathcal{L}^{-1}[F(s)]$	antitrasformata di Laplace	$f(t) = \mathcal{L}^{-1}[F(s)]$
$\mathcal{F}[f(t)]$	trasformata di Fourier	$F(\omega) = \mathcal{F}[f(t)]$
$\mathcal{F}^{-1}[F(\omega)]$	antitrasformata di Fourier	$f(t) = \mathcal{F}^{-1}[F(\omega)]$

continua

Continua tabella C.8

Simbolo	Significato	Note
$\widehat{\mathsf{ABC}}$	angolo	Il vertice è in corrispondenza del punto B
$\overset{\frown}{\mathsf{AB}}$	arco	
$\overline{\mathsf{AB}}$	segmento	
$\vec{V}$	vettore	
$\left\|\vec{V}\right\|, V$	modulo di vettore	
$\vec{V}_1 \cdot \vec{V}_2$	prodotto scalare	Non usare mai l'operatore $\times$ per il prodotto scalare
$\begin{cases}\vec{V}_1 \wedge \vec{V}_2 \\ \vec{V}_1 \times \vec{V}_2\end{cases}$	prodotto vettore	
$\int_\ell \vec{V} \cdot \overrightarrow{\mathrm{d}\ell}$	'lavoro' di $\vec{V}$ lungo la linea ℓ	
$\int_S \vec{V} \cdot \overrightarrow{\mathrm{d}S}$	flusso di $\vec{V}$ attraverso la superficie S	
$\operatorname{grad}\Phi, \nabla\Phi$	gradiente	Si può indicare anche con $\overrightarrow{\operatorname{grad}\Phi}$ oppure $\overrightarrow{\nabla\Phi}$
$\operatorname{div}\vec{V}, \nabla\cdot\vec{V}$	divergenza	
$\operatorname{rot}\vec{V}, \nabla\times\vec{V}$	rotore	Si può indicare anche con $\overrightarrow{\operatorname{rot}\vec{V}}$ oppure $\overrightarrow{\nabla\times\vec{V}}$
$\nabla^2\Phi$	laplaciano di uno scalare	
$\nabla^2\vec{V}$	laplaciano di un vettore	Vettore le cui componenti sono ordinatamente i laplaciani delle componenti di $\vec{V}$. Si può indicare anche con $\overrightarrow{\nabla^2\vec{V}}$

continua

Continua tabella C.8

Simbolo	Significato	Note
x	valore istantaneo	
X	valore efficace	Il concetto ha senso solo se $x(t)$ è periodica
$\hat{x}\,, x_{\max}$	valore massimo	Il valore massimo della funzione $x(t)$ in un intervallo prefissato $t_{\min} \div t_{\max}$
$\check{x}\,, x_{\min}$	valore minimo	Il valore minimo della funzione $x(t)$ in un intervallo prefissato $t_{\min} \div t_{\max}$
$\bar{x}$	valore medio	Il valore medio della funzione $x(t)$ in un intervallo prefissato $t_{\min} \div t_{\max}$
$\mathbf{Re}\, z$	parte reale	
$\mathbf{Im}\, z$	parte immaginaria	Se $z = x + \mathrm{i}\, y$ allora $\mathbf{Im}\, z$ dovrebbe essere uguale a y e non a $\mathrm{i}\, y$, ma talvolta è usato per $\mathrm{i}\, y$
$\lvert z \rvert$	modulo	
$\arg z$	argomento o anomalia	$z = \lvert z \rvert\, \mathrm{e}^{\mathrm{i} \arg z}$
z^*	coniugato	Nei testi matematici è più comune $\bar{z}$
$f_*(s)$	paraconiugato	$f_*(s) = f(-s)$, ma se $f(s)$ è hermitiana, $f_*(\mathrm{i}\, y) = f^*(\mathrm{i}\, y)$
$\mathbf{A}$	insieme	$\mathbf{A} = \{a_1, a_2, \dots\}$. Nello stesso modo, per indicare altri insiemi, si possono usare altre lettere maiuscole, che non siano già associate ad insiemi particolari
$\emptyset$	insieme vuoto	
Ω	universo	
$\mathbf{N}, \mathbb{N}$	insieme dei numeri interi positivi	
$\mathbf{Z}, \mathbb{Z}$	insieme dei numeri interi relativi	
$\mathbf{Q}, \mathbb{Q}$	insieme dei numeri razionali	

continua

Continua tabella C.8

Simbolo	Significato	Note
$\mathbf{R}, \mathbb{R}$	insieme dei numeri reali	
$\mathbf{C}, \mathbb{C}$	insieme dei numeri complessi	
$\mathbf{A} \times \mathbf{B}$	prodotto cartesiano di insiemi	Ogni elemento del prodotto cartesiano è formato dall'accoppiamento di un elemento dell'insieme $\mathbf{A}$ con un elemento dell'insieme $\mathbf{B}$
$\mathbf{R}^n, \mathbb{R}^n$	insieme delle n-uple reali	Indica anche lo spazio reale a n dimensioni
$\mathbf{C}^n, \mathbb{C}^n$	insieme delle n-uple complesse	Indica anche lo spazio complesso ad n dimensioni
$\exists$	esiste	Esiste ed è unico: $\exists!$
$\nexists$	non esiste	
$a \in \mathbf{A}$	appartiene	
$a \notin \mathbf{A}$	non appartiene	
$\mathbf{A} \ni a$	contiene	
$\mathbf{A} \not\ni a$	non contiene	
$\mathbf{A} \cap \mathbf{B}$	intersezione	
$\mathbf{A} \cup \mathbf{B}$	unione	
$\mathbf{A} \backslash \mathbf{B}$	differenza	L'insieme $\mathbf{A} \backslash \mathbf{B}$ è formato dagli elementi di $\mathbf{A}$ esclusi quelli che appartengono anche a $\mathbf{B}$
$\complement_{\Omega} \mathbf{A}$	complemento	$\complement_{\Omega} \mathbf{A} = \Omega \backslash \mathbf{A}$
$\mathbf{A} \subset \mathbf{B}$	è contenuto	$\mathbf{A}$ è un sottoinsieme di $\mathbf{B}$
$\mathbf{A} \not\subset \mathbf{B}$	non è contenuto	
$\mathbf{A} \subseteq \mathbf{B}$	è contenuto o coincide	
$\mathbf{A} \not\subseteq \mathbf{B}$	non è contenuto né coincide	

continua

Continua tabella C.8

Simbolo	Significato	Note
$\mathbf{B} \supset \mathbf{A}$	contiene	**B** contiene l'insieme **A**
$\mathbf{B} \not\supset \mathbf{A}$	non contiene	
$\mathbf{B} \supseteq \mathbf{A}$	contiene o coincide	
$\mathbf{B} \not\supseteq \mathbf{A}$	non contiene né coincide	
$\boldsymbol{A}$	matrice	Quando la matrice ha una sola colonna (riga) si è soliti usare lettere minuscole. La matrice può essere esplicitata in uno dei modi seguenti $\boldsymbol{A} = \begin{pmatrix} a_{11} & \dots & a_{1r} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nr} \end{pmatrix}$ oppure $\boldsymbol{A} = \begin{bmatrix} a_{11} & \dots & a_{1r} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nr} \end{bmatrix}$
$\lvert\boldsymbol{A}\rvert, \det \boldsymbol{A}$	determinante	La matrice $\boldsymbol{A}$ di cui si calcola il determinante deve essere quadrata. È $\det \boldsymbol{A} = \begin{vmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{vmatrix}$
$\lVert\boldsymbol{A}\rVert$	norma	
$\operatorname{tr} \boldsymbol{A}$	traccia	$\operatorname{tr} \boldsymbol{A} = \sum_{i=1}^{n} a_{ii}$
$\boldsymbol{A}^*$	matrice coniugata	Nei testi di matematica è più frequente $\bar{\boldsymbol{A}}$
$\tilde{\boldsymbol{A}}, {}^t\boldsymbol{A}$	matrice trasposta	Si indica anche con $\boldsymbol{A}^{\mathrm{T}}$. Questo secondo simbolo è più adatto quando la matrice è esplicitata mediante i suoi elementi
$\bar{\boldsymbol{A}}$	matrice associata	$\bar{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^*$
$\boldsymbol{A}_*$	matrice paraconiugata	$\boldsymbol{A}_*(s) = \tilde{\boldsymbol{A}}(-s)$, ma se gli elementi $a_{ij}(s)$ di $\boldsymbol{A}$ sono hermitiani $\forall\, i,j$, è $\boldsymbol{A}_*(\mathrm{i}\, y) = \bar{\boldsymbol{A}}(\mathrm{i}\, y)$

continua

Continua tabella C.8

Simbolo	Significato	Note
$\boldsymbol{D}$	matrice diagonale	$\boldsymbol{D} = \boldsymbol{diag}\,(a_1,\ldots,a_n)$. Può essere usata qualunque altra lettera, purché ne sia definito il significato
$\boldsymbol{I}$, $\boldsymbol{1}$	matrice identità	Il secondo simbolo può essere usato solo quando il primo possa ingenerare confusione con altre grandezze
$\boldsymbol{\mathcal{G}}$	diadica	
$s_{\overline{n}\,i}$	capitalizzazione	Coefficiente di capitalizzazione di n annualità posticipate all'interesse i
$\ddot{s}_{\overline{n}\,i}$	capitalizzazione	Coefficiente di capitalizzazione di n annualità anticipate all'interesse i
$a_{\overline{n}\,i}$	attualizzazione	Coefficiente di attualizzazione di n annualità posticipate all'interesse i
$\ddot{a}_{\overline{n}\,i}$	attualizzazione	Coefficiente di attualizzazione di n annualità anticipate all'interesse i
$\sigma_{\overline{n}\,i}$	reintegrazione	Coefficiente di reintegrazione mediante n annualità posticipate all'interesse i
$\ddot{\sigma}_{\overline{n}\,i}$	reintegrazione	Coefficiente di reintegrazione mediante n annualità anticipate all'interesse i
$\alpha_{\overline{n}\,i}$	ammortamento	Coefficiente di ammortamento mediante n annualità posticipate all'interesse i
$\ddot{\alpha}_{\overline{n}\,i}$	ammortamento	Coefficiente di ammortamento mediante n annualità anticipate all'interesse i

Tabella C.9: Nomenclatura, simboli e unità di misura

Grandezza	**Simbolo**	**Unità SI**
angolo piano	$\alpha, \beta, \gamma, \ldots$	rad
angolo solido	ω, Ω	sr
lunghezza	l	m
larghezza	b	m
altezza	h	m
raggio	r	m
spessore	d, δ	m
diametro	d	m
percorso curvilineo	s	m
superficie, area	S, A	m^2
volume	V, v	m^3
lunghezza d'onda	λ	m, (m/onda)
numero d'onda $(1/\lambda)$	σ	m^{-1}, (onde/m)
ondulanza $(2\pi/\lambda)$	k	m^{-1}
attenuazione spaziale	α	m^{-1}, (Np/m)
costante di fase	β	m^{-1}
costante di propagazione $(\alpha + \mathrm{i}\,\beta)$	γ	m^{-1}
tempo	t	s
periodo	T	s, (s/ciclo)
frequenza	f	Hz, (cicli/s)
pulsazione	ω	s^{-1}
tempo di rilassamento o costante di tempo	τ	s, (s/Np)

continua

Continua tabella C.9

Grandezza	**Simbolo**	**Unità SI**
coefficiente di smorzamento	δ	s^{-1}, (Np/s)
decremento logaritmico (T/τ)	Λ	(Np/ciclo)
velocità	v, u	m/s
velocità angolare	ω	rad/s
accelerazione	a	$\mathrm{m/s^2}$
accelerazione angolare	α	$\mathrm{rad/s^2}$
accelerazione di gravità	g	$\mathrm{m/s^2}$
costante di gravitazione	G	$\mathrm{N\,m^2/kg^2}$
velocità della luce nel vuoto	c_0	m/s
massa	m	kg
massa volumica	ϱ	$\mathrm{kg/m^3}$
densità relativa (all'acqua)	d	–
volume massico ($1/\varrho$)	v	$\mathrm{m^3/kg}$
quantità di moto	p	kg m/s
momento della quantità di moto	L	$\mathrm{kg\,m^2/s}$
momento quadratico di superficie	I	$\mathrm{m^4}$
momento di inerzia	J	$\mathrm{kg\,m^2}$
forza	F	N
coppia	T, M	N m, (N m/rad)
momento di una forza	M	N m, (N m/rad)
pressione	p	Pa
tensione normale	σ	Pa
tensione di taglio	τ	Pa

continua

Continua tabella C.9

Grandezza	**Simbolo**	**Unità SI**
allungamento relativo	ε	–
modulo di elasticità	E	Pa
angolo di torsione	γ	rad
modulo di torsione	G	Pa
dilatazione volumica relativa	ϑ	–
modulo di compressione	K	Pa
rapporto di Poisson	μ	–
viscosità dinamica	η	Pa s
viscosità cinematica (η/ϱ)	ν	m^2/s
coefficiente di attrito	μ	–
tensione superficiale	γ, σ	N/m
energia	E	J
energia potenziale	E_p, V, Φ	J
energia cinetica	E_k, T, K	J
lavoro	W	J
potenza	P	W
rendimento	η	–
velocità del suono	c	m/s
velocità longitudinale	c_l	m/s
velocità trasversale	c_t	m/s
velocità di gruppo	c_g	m/s
flusso energetico (acustico)	P	W/m^2
fattore di riflessione (acustica)	ϱ	–

continua

Continua tabella C.9

Grandezza	**Simbolo**	**Unità SI**
fattore di assorbimento (acustico)	$\alpha_\mathrm{a}, \alpha$	–
fattore di trasmissione (acustica)	τ	–
fattore di dissipazione (acustica)	δ	–
livello sonoro	L_N, Λ	dB
corrente elettrica	i, I	A
densità di corrente	j, J	$\mathrm{A/m^2}$
carica elettrica	Q	C
densità volumica di carica	ϱ	$\mathrm{C/m^3}$
densità superficiale di carica	σ	$\mathrm{C/m^2}$
potenziale elettrico	V	V
tensione (elettrica)	V	V
impulso di tensione	U	V s
forza elettromotrice	E	V
campo elettrico	E, K	V/m
spostamento elettrico	D	$\mathrm{C/m^2}$
flusso elettrico	Ψ	C
capacità	C	F
permittività (o permettività)	ε	F/m
permittività del vuoto	ε_0	F/m
permittività relativa	ε_r	–
polarizzazione dielettrica	P	$\mathrm{C/m^2}$
suscettività elettrica $(\varepsilon_\mathrm{r} - 1)$	χ_e	–
elettrizzazione $(D/\varepsilon_0 - E)$	$E_\mathrm{i}, K_\mathrm{i}$	V/m

continua

Continua tabella C.9

Grandezza	**Simbolo**	**Unità SI**
polarizzazione ($D - \varepsilon_0 E$)	P	$\mathrm{C/m^2}$
momento di dipolo (elettrico)	p	$\mathrm{C\,m}$
campo magnetico	H	$\mathrm{A/m}$
potenziale magnetico	U_m	A
forza magnetomotrice	F_m	A
induzione magnetica	B	T
flusso di induzione (magnetica)	Φ	Wb
permeabilità	μ	$\mathrm{H/m}$
permeabilità del vuoto	μ_0	$\mathrm{H/m}$
permeabilità relativa	μ_r	–
magnetizzazione	M	$\mathrm{A/m}$
suscettività magnetica ($\mu_\mathrm{r} - 1$)	χ_m, κ	–
momento elettromagnetico	m, μ	$\mathrm{A\,m^2}$
polarizzazione magnetica	J	T
resistenza	R	Ω
reattanza	X	Ω
impedenza	Z	Ω
fattore di qualità	$Q_\mathrm{L}, Q_\mathrm{C}, \ldots$	–
coefficiente di risonanza	Q	–
conduttanza	G	S
suscettanza	B	S
ammettenza	Y	S
resistività	ϱ	$\Omega\,\mathrm{m}$

continua

Continua tabella C.9

Grandezza	**Simbolo**	**Unità SI**
conducibilità	σ, γ	S/m
induttanza (propria)	L	H
induttanza mutua	M	H
coefficiente di accoppiamento ($M/\sqrt{L_\mathrm{p}L_\mathrm{s}}$)	k	–
coefficiente di dispersione ($1-k^2$)	σ	–
riluttanza	$\mathcal{R}, R$	H^{-1}
permeanza	Λ	H
potenza reattiva	Q	V A
potenza apparente	P	V A
sfasamento	φ	rad
numero delle fasi	m	–
angolo di perdita	δ	rad
numero di spire	N, n	–
densità volumica di energia elettromagnetica	w	$\mathrm{J/m^3}$
vettore di Poynting	S	$\mathrm{W/m^2}$
potenziale vettore magnetico	$\mathcal{A}, A$	Wb/m
temperatura termodinamica	T	K
temperatura (Celsius)	t	°C
quantità di calore	Q	J
entropia	S	J/K
energia interna	U	J
energia libera ($U-TS$)	F	J
entalpia	H	J

continua

Continua tabella C.9

Grandezza	**Simbolo**	**Unità SI**
entalpia libera	G	J
coefficiente di pressione $(\partial(\ln p)/\partial T\|_V)$	β	K^{-1}
compressibilità $(-\partial(\ln V)/\partial p\|_T)$	κ	m^2/N
coefficiente di dilatazione lineare	α	K^{-1}
coefficiente di dilatazione volumica	γ	K^{-1}
conducibilità termica	λ	$\mathrm{W}/(\mathrm{m\,K})$
calore massico	c_p, c_v	$\mathrm{J}/(\mathrm{kg\,K})$
capacità termica	C_p, C_v	J/K
rapporto dei calori massici	κ	–
flusso termico	Φ	W
flusso di calore areico	q	W/m^2
coefficiente di trasmissione termica	τ	$\mathrm{W}/(\mathrm{m}^2\mathrm{K})$
coefficiente di diffusione termica	a	m^2/s
potenza raggiante	Q, W	W
intensità energetica	I	W/sr
irradiamento	E	W/m^2
radianza	L	$\mathrm{W}/(\mathrm{m}^2\,\mathrm{sr})$
intensità luminosa	I	cd
flusso luminoso	Φ	lm
quantità di luce	Q	$\mathrm{lm\,s}$
luminanza	L	cd/m^2
illuminamento	E	lx
fattore di assorbimento	α	–

continua

Continua tabella C.9

Grandezza	**Simbolo**	**Unità SI**
fattore di riflessione	ϱ	–
fattore di trasmissione	τ	–
indice di rifrazione	n	–
distanza di due piani reticolari adiacenti	d	m
angolo di Bragg	ϑ	rad
massa efficace dell'elettrone	m^*, m_{eff}	kg
energia di Fermi	E_{F}	J
vettore d'onda di Fermi	k_{F}	m^{-1}, (rad/m)
coefficiente di Seebeck	S	V/K
coefficiente di Peltier	Π	V
coefficiente piezoelettrico (polarizzazione/sforzo)	d_{mn}	C/N
temperatura caratteristica di Weiss	Θ_{W}	K
temperatura di Curie	T_{C}	K
temperatura di Neel	T_{N}	K
coefficiente di Hall	R_{H}	$\mathrm{V\,m^2/A^2}$
quantità di sostanza	n	mol
massa molare	M	kg/mol
volume molare	V_{m}	$\mathrm{m^3/mol}$
energia interna molare	U_{m}	J/mol
capacità termica molare	$C_{p\mathrm{m}}, C_{v\mathrm{m}}$	$\mathrm{J/(mol\,K)}$
entropia molare	S_{m}	$\mathrm{J/(mol\,K)}$
concentrazione di un costituente	c	$\mathrm{mol/m^3}$
molalità di un soluto	m	mol/kg

continua

Continua tabella C.9

Grandezza	**Simbolo**	**Unità SI**
dose assorbita	D	Gy
energia impartita massica	z	Gy
indice di dose assorbita	D_{i}	Gy
rateo di dose assorbita	$\dot{D}$	Gy/s
equivalente di dose	H	Sv
esposizione	X	C/kg
rateo di esposizione	$\dot{X}$	C/(kg s)
attività di un radio nuclide	A	Bq

Appendice D

Divisione in sillabe

L'algoritmo per dividere in sillabe le parole in fin di riga è una specialità del sistema TEX, ed è stata incorporata in diversi altri programmi, tra i quali merita segnalare OpenOffice come esempio di software Open Source, a cui possono collaborare tutti gli utenti che ne hanno la possibilità e la competenza; un po' come succede per il sistema TEX.

Prima di tutto bisogna capire come fa TEX a individuare una parola, ciò che il programma crede che sia una parola; per TEX una parola è una sequenza di lettere maiuscole o minuscole che comincia dopo uno spazio ma può essere preceduta da segni di interpunzione in senso lato (per esempio: parentesi, virgolette, ecc.) e finisce con il primo token che non ha le caratteristiche di una lettera. Questa stringa non può essere seguita da token diversi da una lettera che appartengano a certe categorie, per esempio non può essere seguita dal comando `\footnote`. Non può nemmeno essere preceduta da una macro, perché lo spazio che separa la macro dalla stringa di lettere serve già all'interprete TEX per capire che la macro è terminata; la stringa può però costituire l'argomento di una macro.

Il lettore può sfruttare il comando `\showhyphens` per vedere come TEX divida in sillabe; usando **babel** con l'opzione per l'italiano potrebbe vedere che cosa succede scrivendo;

```
\showhyphens{macroistruzione macro"istruzione}
```

e troverebbe sullo schermo o nel file log un messaggio del tipo

```
Underfull \hbox (badness 10000) in paragraph at lines 5300--5300
[] \fontiii ma-croi-stru-zio-ne ma-cro-istru-zio-ne
```

A seconda del sistema operativo e della particolare distribuzione del sistema TEX, il nome `\fontiii` potrebbe essere sostituito da qualcosa d'altro; in ogni caso esso rappresenta la descrizione interna del font in uso nel momento in cui viene eseguito il comando `\showhyphens`.

L'annotazione 'Underfull hbox (badness 10000) in paragraph at lines ...' può venire ignorata; per altro è il messaggio che TEX emette ogni volta che deve allargare troppo lo spazio interparola per giustificare una riga.

La parte più interessante viene dopo; infatti 'ma-croi-stru-zio-ne' è la divisione in sillabe che TEX eseguirebbe spontaneamente, senza sapere che si tratta di

una parola composta, mentre ‘ma-cro-istru-zio-ne’ è la divisione in sillabe conseguente all’uso del carattere attivo `"`, che, come si è già detto, serve appunto (tra l’altro) per indicare una divisione etimologica, piuttosto che fonetica.

Si noti che la potenziale parola talvolta non coincide con quello che gli umani sanno essere una parola della loro lingua o di un’altra lingua straniera.

Se si usasse la codifica OT1, invece della codifica T1, gli accenti vengono collocati sulle vocali (o sopra o sotto altre lettere, come per esempio: ç oppure č in molte altre lingue diverse dall’italiano) mediante un comando primitivo che si chiama `\accent`; quindi anche se uno scrive `qualità`, quando TEX interpreta il contenuto del file sorgente, sostituisce la ‘à’ con una sequenza di istruzioni, in particolare sostituisce la stringa iniziale con `qualit\accent18a`; in questo modo per TEX quello che lui crede essere una parola è la stringa ‘qualit’; all’interno di questa stringa esso trova un solo punto di divisione, ‘qua-lit’ e quindi l’effetto complessivo sarebbe di dividere la parola in ‘qua-lità’ invece che in ‘qua-li-tà’.

Queste cesure mancate potrebbero essere talvolta causa di composizioni in cui qualche riga sporge fuori dalla giustezza, oppure qualche riga è stata spaziata troppo per consentire la giustificazione; tuttavia il problema è facilmente risolubile; basta usare la direttiva

```
\usepackage[T1]{fontenc}
```

come si è già detto diverse volte.

Quando si esamina il file log per cercare di capire perché qualche cesura in fin di riga ha provocato righe troppo lunghe o troppo spaziate, si scopre sempre che le cause sono quelle descritte sopra: una macro precede troppo da vicino o segue troppo da vicino una stringa di lettere che potrebbe essere una parola del linguaggio; oppure si è usato un font che non consente la cesura, come per esempio un font della famiglia a spaziatura fissa.

Un espediente elementare sarebbe quello di dividere a mano: basta inserire il comando `\-` all’interno di una parola nel punto in cui si vorrebbe eseguire la cesura; in questo testo è stato eseguito solo per una voce dell’indice analitico formata da una parola talmente lunga che invadeva la colonna adiacente.

Ma più in generale basta inserire una spaziatura orizzontale di ampiezza nulla fra la parola e l’oggetto ‘troppo vicino’, come per esempio una macro o una nota. Ciò si ottiene usando comandi di basso livello come per esempio

```
\hskip 0pt
```

dove `\hskip` è il comando primitivo che inserisce lo spazio orizzontale specificato immediatamente dopo, nel nostro caso 0 pt.

È bene conoscere questo meccanismo al fine di porre rimedio alle rare circostanze in cui il meccanismo di sillabazione non svolge il suo compito come ci si aspetterebbe.

Vale la pena di aggiungere una osservazione. Nessun meccanismo automatico è infallibile; la sillabazione in italiano funziona molto bene, ma non si può escludere che un neologismo particolare o una parola tecnica possa risultare divisa in sillabe in modo scorretto, specialmente se è ricavato da una radice straniera. Per esempio la parola tecnica dell’ambito medico ‘dispepsia viene divisa in ‘di-spep-sia’; ovviamente questo è un caso semplice da trattare mediante l’uso del carattere attivo `"`, perché scrivendo `dis"pepsia`, la parola risulta divisa in modo etimologico. Dalla chimica si potrebbe prendere un lunga parola composta, per esempio *desclorfeniraminacloridrato* che può

venire scritta nella forma *des"clor"fenir"amina"clor"idrato* che porta sempre alla divisione in sillabe corretta, anche nei punti non esplicitamente indicati: *des-clor-fe-nir-ami-na-clor-idra-to*.

In inglese il meccanismo di sillabazione sembra funzionare correttamente in poco più del 90% dei casi. In francese l'algoritmo funzionerebbe benissimo, nonostante le numerose lettere con diacritici, ma le regole della buona educazione impediscono di dividere in sillabe parole che lascino nella riga contenente la cesura, o nella riga seguente, monconi di parole che hanno significati volgari; a causa di ciò alcune parole non vengono divise affatto o vengono divise in monconi talvolta troppo lunghi. In tedesco il problema è veramente difficile a causa delle numerosissime parole composte che vanno divise etimologicamente, talvolta con un cambio di ortografia. Tuttavia l'algoritmo viene usato con soddisfazione da tutti, e nei pochi casi in cui fallisce ci sono almeno due strade assai semplici da seguire:

1. Si marcano a mano con `\-` i punti di divisione di quelle poche parole che nelle bozze risultano divise in modo scorretto; con l'italiano ciò è estremamente raro, per non dire impossibile, perché le norme UNI che regolano la materia ammettono come corrette tutte le divisioni fonetiche; certo con parole straniere o derivate da radici straniere questo potrebbe non essere corretto.

2. Si usa per l'italiano il carattere attivo `"` nelle parole composte che si desidera dividere etimologicamente; si noti che le norme UNI che regolano questa questione in italiano accettano la divisione fonetica anche per le parole composte, ma in uno scritto tecnico è preferibile usare la divisione etimologica. Nelle altre lingue straniere, tranne in inglese, le opzioni di **babel** prevedono quasi sempre un comando che utilizza il carattere attivo`"`, per esempio `"-`, che consente di svolgere la stessa funzione che `"` da solo svolge in italiano; bisogna consultare la parte di documentazione di **babel** relativa alla lingua specifica.

3. Si ricorre alla lista di parole divise in sillabe contenute nell'argomento del comando `\hyphenation`; per l'italiano, per esempio, se si dovesse usare spesso la parola 'macroistruzione' al singolare e al plurale basterebbe scrivere nel preambolo:

   ```
   \hyphenation{ma-cro-istru-zio-ne ma-cro-istru-zio-ni}
   ```

   Nel preambolo si possono usare diverse liste introdotte mediante il comando `\hyphenation`; i loro contenuti vengono raccolti tutti assieme in una apposita memoria del programma, il quale consulta sempre questa lista prima di attivare l'algoritmo di sillabazione vero e proprio.

   Con lingue che usano la declinazione (come succede in parte per l'italiano) e la coniugazione, queste liste potrebbero diventare immense. In italiano un sostantivo richiede due voci: singolare e plurale; per un aggettivo ne possono essere richieste quattro: singolare e plurale, maschile e femminile; per un verbo occorrono una sessantina di voci. In italiano si ricorre assai raramente a questo meccanismo, ma, per esempio, per l'inglese ogni installazione del sistema TEX prevede la lettura automatica iniziale di una lista di eccezioni di diverse centinaia di parole.

.d	e	l	l	’	i	s	t	r	u	z	i	o	n	e.
.d	0 e	0 l	0 l	0 ’	0 i	0 s	0 t	0 r	0 u	0 z	0 i	0 o	0 n	0 e.
		1 l		’ 2		1 s	2	1 r		1 z			1 n	
			1 l				1 t							
		2 l	0 l				t	2 r						
.d	0 e	2 l	1 l	0 ’	2 i	1 s	2 t	2 r	0 u	1 z	0 i	0 o	1 n	0 e.

Tabella D.1: Pattern usati per dividere in sillabe *dell’istruzione*

Tuttavia, superati questi punti, vediamo come fa TEX a dividere in sillabe.

Esso spezza ogni parola che si trovi verso la fine di ogni riga nelle sue componenti di una sola lettera, di due lettere, di tre lettere, ma con un procedimento sequenziale che non comporti tutte le possibili scomposizioni; nel fare questo esso è guidato dai ‘pattern di sillabazione’. Questi sono contenuti in un file che in fase di costruzione del formato viene allocata in una struttura dati molto veloce da esplorare; è per questo che bisogna sempre essere sicuri che detto file sia stato elaborato nella costruzione del formato, altrimenti la sillabazione per la lingua cui corrispondono quei pattern è impossibile. Essi sono formati da monconi di parole in cui è indicata la possibilità di dividere in sillabe mediante delle cifre pari (non dividere) o dispari (dividi pure) che possono andare da 0 a 9. Se fra le stesse due lettere di pattern diversi compaiono cifre diverse, non si tratta di una contraddizione, ma fra le due indicazioni prevale quella con la cifra più alta. Per esempio, il pattern `1b` dice che si può dividere prima della ‘b’, ma non dopo (la cifra assente implica la cifra zero); ma il pattern `2bb` dice che si può andare a capo *dopo* la prima ‘b’, ma *non* prima. Infatti, tenendo conto dei due pattern, il secondo si legge come se fosse `2b1b0`, il che conferma quanto appena descritto a parole.

Prendiamo un esempio più complesso, per esempio la ‘parola’ con elisione e apostrofo: *dell’istruzione*. Conviene rifarsi alla tabella D.1; in questa tabella la parola iniziale è scomposta nelle sue componenti ed è preceduta e seguita da due segni di punto; questi rappresentano per TEX i delimitatori della stringa di lettere che lui ritiene essere una parola. Nella seconda riga tutti i codici di cesura pari al valore di default ‘zero’ sono intercalati alle lettere. Nella terza e nella quarta riga sono indicati, traendoli dal file `ithyph.tex`, i pattern che coinvolgono una sola lettera, contornati da rispettivi codici di consenso o interdizione della cesura; come si vede questi pattern coinvolgono solo le consonanti e l’apostrofo. Nella quinta riga sono riportati i pattern che coinvolgono due lettere insieme ai rispettivi codici di consenso e interdizione della cesura; il file `ithyph.tex` non mostra altri pattern di tre o più lettere che si possano trovare nella parola indicata, qunidi non ci sono altre righe da scrivere nella tabellina. Perciò nell’ultima riga, oltre alle lettere componenti, sono riportate le cifre più alte che compaiono in ogni colonna di cifre. Ne consegue che le cesure possono cadere solo fra le due ‘l’, tra la ‘i’ e la ‘s’, tra la ‘u’ e la ‘z’ e tra la ‘o’ e la ‘n’, dove compaiono i codici dispari.

La tabella D.2 mostra un esempio ancora più complesso; non lo si commenta, perché il procedimento esposto in relazione alla tabella D.1 è il medesimo. Questa volta la parola da dividere in sillabe è *discinesia*, che i medici pronunciano con il prefisso ‘dis’ separato dalla parola ‘cinesia’. Il risultato infatti porta alla divisione etimologicamente corretta *dis-ci-ne-sia*. Il lungo pattern che compare

.d	i	s	c	i	n	e	s	i	a.
.d0i0s0c0i0n0e0s0i0a.									
		1s2			1n		1s2		
			1c						
.d0i2s3c0i0n0e									
.d0i2s3c0i1n0e1s2i0a.									

Tabella D.2: Pattern usati per dividere in sillabe *discinesia*

nella quinta riga, serve appunto per specificare l'eccezione alla regola di non separare la 's' impura (che in questo caso, insieme alla 'c' e alla 'i' forma un 'trigrafo' che si pronuncia diversamente dalle singole lettere componenti); il pattern è sufficientemente lungo per isolare la radice di 'discinesia' e dei suoi derivati, come per esempio 'discinetico', ma non coinvolge altre parole che cominciano nello stesso modo, per esempio 'disciplina, 'discinto', eccetera.

Per l'inglese i pattern sono quasi 5000, le eccezioni introdotte con il comando `\hyphenation` sono alcune centinaia e la probabilità di una divisione errata è bassa, ma non nulla.

In italiano i pattern sono circa 330, la lista delle eccezioni è nulla e negli ultimi 10 anni non sono stati segnalati errori; tutto ciò la dice lunga sulla facilità della divisione fonetica dell'italiano e della difficoltà della divisione ritmica dell'inglese; in questa lingua due parole ortograficamente identiche ma che si pronunciano con accenti ritmici diversi, richiedono divisioni diverse: *record* (nome) e *record* (verbo) si dividono rispettivamente in *rec-ord* e *re-cord*!

Esercizio D.1 Come già detto, i pattern per l'italiano sono contenuti nel file `ithyph.tex`; il lettore cerchi nell'elenco dei pattern quelli che corrispondono alle possibili divisioni di una parola a sua scelta e controlli a mano la divisione prodotta dai pattern; successivamente inserisca la stessa parola dentro al comando `\showhyphens` e verifichi se LaTeX divide come egli ha trovato a mano. Per semplicità usi la classe *`minimal`*.

Esercizio D.2 Perché la lista dei pattern contiene l'attribuzione di un codice strano all'apostrofo e poi la lista dei pattern contiene anche pattern con uno o due apostrofi?

Esercizio D.3 Il file dei pattern non contiene nulla per dividere in sillabe in modo etimologico la parola *dispepsia*, anche perché il dizionario Garzanti indica solo la divisione *di-spep-sia*.

Quale pattern bisognerebbe aggiungere a quelli esistenti per dividere questa parola in modo etimologico?

In alcune circostanze, pur dividendo in sillabe le parole in fin di riga, rimangono delle righe sporgenti fuori della giustezza oppure troppo 'brutte' perché lo spazio interparola è stato allargato troppo.

Non è il caso di preoccuparsi troppo se l'indice di bruttezza (*badness* in inglese) è di qualche centinaio o di poche migliaia; il messaggio di bruttezza, che viene esposto sullo schermo e trascritto nel file `.log`, comincia a diventare preoccupante se si supera il valore 5000, ma certamente la bruttezza è 'infinita' se il messaggio riporta una bruttezza di 10 000.

Per la bruttezza delle righe si può specificare nel preambolo un limite sotto al quale il programma non deve preoccuparsi; il parametro `\hbadness` può venire impostato nel preambolo ad un valore diverso a seconda del tipo di documento e del suo contenuto; per esempio

```
\hbadness=5000
```

permette di eliminare tutti i messaggi relativi alle righe la cui bruttezza sia inferiore al valore 5000.

In modo analogo si possono eliminare alcuni messaggi per la bruttezza verticale, che TEX emette quando deve allargare troppo la gomma dei contrografismi verticali, durante la composizione della pagina; il parametro specifico si chiama `\vbadness`; talvolta la bruttezza verticale può essere notevole, ma, a parte stabilire un valore per il parametro appena menzionato, se si vuole rendere la composizione veramente professionale bisogna apportare modifiche al testo da comporre. Generalmente questa bruttezza verticale dipende dal fatto che un oggetto ingombrante (come ad esempio il titolino di una sezione seguito da almeno due righe di testo) non trova posto al fondo di una pagina. Allungando il testo che prcede l'oggetto ingombrante si riduce l'ammontare dell'allargamento della gomma verticale.

Tornando alle righe brutte si può procedere come per le pagine brutte: si modifica il testo. Tuttavia sarebbe meglio conoscere il meccanismo con cui TEX divide i capoversi in righe. Siccome però questo argomento si addentra nelle parti più tecniche, si rinvia il lettore al testo di base, il TEXbook, dove l'argomento è sviscerato nei dettagli.

Qui il lettore si accontenti di sapere che la brutta divisione che TEX ha trovato è quella che rende minima la bruttezza dell'intero capoverso; dunque c'è poco da fare per ridurre la bruttezza, visto che questa è la minima possibile. Si può tentare, per esempio, di introdurre degli altri punti di cesura mediante il comando `\-`.

Ci si ricordi, inoltre, che i pattern per l'italiano non dividono i dittonghi (ed è giusto), ma non dividono nemmeno gli iati; questa decisione è stata presa con una motivazione 'psicologica': il lettore si trova a disagio se trova un gruppo di due o più vocali spezzato fra due righe, anche quando la grammatica lo consentirebbe; si è deciso di spezzare i gruppi di tre o più vocali se queste contengono almeno un dittongo o un 'trittongo', ma lasciando un eventuale trittongo indiviso; quindi 'quieto' si divide solo in 'quie-to' perché contiene un trittongo, ma 'maieutica' si divide in 'ma-ieu-ti-ca' perché le quattro vocali contengono un trittongo. Però 'aiuola' viene diviso solamente in 'aiuo-la' e non in 'a-iuo-la', come la grammatica consentirebbe, perché TEX non divide dopo sillabe iniziali o prima di sillabe finali 'troppo corte'.

Una sillaba iniziale o finale è troppo corta se contiene meno lettere del numero specificato dai parametri `\lefthyphenmin` e `\righthyphenmin`; per l'italiano entrambi questi parametri valgono 2, mentre per l'inglese, il francese ed altre lingue essi valgono rispettivamente 2 e 3.

Dunque per un dato capoverso diviso in modo brutto, si potrebbe formare un gruppo dove si impostano i due parametri a valori più bassi e/o si inseriscono le divisioni esplicite, per esempio, di `pa\-e\-se` oppure di `i\-de\-a\-le`. La bruttezza si sposta dalla forma del capoverso alla cattiva qualità delle sillabe.

Un'altra possibilità è quella di usare l'ambiente *sloppypar* con le impostazioni e la sintassi seguente:

```
\begin{sloppypar}\tolerance=⟨tolleranza⟩
⟨capoverso da comporre⟩
\end{sloppypar}
```

Il valore ⟨*tolleranza*⟩ viene specificato mediante un numero inferiore a 10 000, ma altrimenti piuttosto alto; chi scrive solitamente usa il valore 9999. Si tratta di una forte tolleranza, che però, secondo TEX, non è infinita. Il capoverso avrà una certa bruttezza, ma non sarà infinitamente brutto.

La soluzione migliore, se il capoverso lo consente, è però quella di modificare il testo; spesso basta aggiungere o togliere una parola, oppure scambiare di posto due parole, oppure riformulare un periodo o una frase. Se il capoverso non è troppo corto le possibilità sono innumerevoli.

Appendice E

Riepilogo della sintassi di LaTeX

In questa appendice si riepilogherà in modo molto succinto la sintassi di tutti (o quasi) i comandi LaTeX, anche quelli non descritti nei capitoli del testo. L'ordine del materiale segue l'appendice C del manuale di Lamport, il creatore di LaTeX. Non si tratta però di una semplice traduzione, anche se la materia sembra esposta nello stesso ordine; è una esposizione ragionata e commentata, anche alla luce di informazioni rese disponibili dopo la pubblicazione di quel manuale.

Parentesi graffe e quadre Le parentesi graffe generalmente racchiudono o gli argomenti obbligatori dei comandi oppure un insieme di testo e di comandi e istruzioni che formano un gruppo; i comandi e le istruzioni mantengono la loro efficacia solo all'interno del gruppo, tranne alcuni comandi il cui effetto è globale; questi comandi sono:

`\newcounter`	`\pagenumbering`	`\newlength`	`\hyphenation`
`\setcounter`	`\thispagestyle`	`\newsavebox`	
`\addtocounter`	`\pagecolor`	`\newtheorem`	

Le parentesi quadre delimitano gli argomenti facoltativi di molti comandi; bisogna stare attenti alle possibili contraddizioni fra queste parentesi e le parentesi quadre che compaiono nel testo; queste potrebbero essere scambiate per quelle. Al fine di evitare queste incomprensioni è opportuno racchiudere le parentesi quadre testuali (o le frasi racchiuse fra parentesi quadre) fra parentesi graffe. I comandi con cui potrebbero succedere queste incomprensioni sono i seguenti:

`\\`	`\linebreak`	`\nolinebreak`	`\newcounter`	`\twocolumn`
`\item`	`\pagebreak`	`\nopagebreak`	`\newtheorem`	`\suppressfloats`

Si scriverà per esempio

```
\begin{itemize}
\item {[omissis]} il conte disse allora: "<Ah, ah!">
\item[{[omissis]}] serve per indicare
                  l'omissione di una parte di testo.
...
```

Gli argomenti degli ambienti Quando si apre un ambiente, solo il comando di apertura può ricevere degli argomenti, mentre il comando di chiusura non ne può accettare alcuno. Se l'ambiente ha un nome che contiene un asterisco finale, questo va ripetuto anche nel comando di chiusura; l'asterisco non è un argomento, ma fa parte integrante del nome.

Comandi fragili e robusti Quasi tutti i comandi di LaTeX sono robusti, nel senso che eseguono quel che devono eseguire senza che questa operazione possa essere modificata dalla posizione che essi hanno, per esempio negli argomenti di altri comandi. Sono fragili gli altri, e per evitare che possano combinare pasticci, bisogna proteggerli con `\protect` premesso al loro nome. Il LaTeX 3 Team sta facendo l'impossibile per eliminare e/o modificare i pochi comandi fragili che rimangono, tuttavia ce ne sono ancora alcuni.

I comandi fragili si possono comportare male quando sono contenuti fra gli argomenti mobili di altri comandi; sono argomenti mobili quelli che devono venire elaborati in fasi successive, per esempio quando devono essere scritti nei file ausiliari e poi riletti ed elaborati in una fase successiva quando si rilancia LaTeX.

I comandi con argomenti mobili sono tipicamente quelli che scrivono qualcosa che deve poi comparire negli indici generale e/o analitico; nei glossari, nelle liste delle figure e/o delle tabelle, eccetera; gli argomenti di `\footnote` e di `\footnotetext`; gli argomenti di `\typein` e di `\typeout` che consentono di creare dei file `.tex` interattivi; il contenuto dell'ambiente *letter* definito all'interno della classe *letter*; l'argomento di `\thanks` usato tipicamente nei titoli, specialmente per indicare le afferenze degli autori; l'argomento facoltativo di `\bibitem`; l'argomento delle *@-espressioni* nella definizione degli incolonnamenti degli ambienti *tabular* e *array*.

È opportuno usare `\protect` con parsimonia, per non correre il rischio di inserirlo dove proprio non ci vuole; conviene inserirlo a posteriori quando l'esecuzione della compilazione mostra errori irrecuperabili, spesso di difficile diagnosi, ma altrettanto spesso dovuti alla fragilità di qualche comando.

**Il comando ** Il comando `\\` serve sia per andare a capo durante la composizione del testo normale, eventualmente contenuto dentro ambienti che specificano il tipo di composizione (testi in display), sia nella composizione di tabelle o matrici, quando si vuole terminare una riga di quella struttura. I due usi possono entrare in collisione, specialmente nelle colonne definite con i descrittori di colonna `p`, `m` o `b`.

Quando LaTeX compone un capoverso si eviti di usare due comandi `\\` di seguito; si usi invece l'argomento facoltativo per aggiungere altro spazio; la sintassi del comando, lo si ricorda, è:

```
\\{⟨spazio⟩}
\\*{⟨spazio⟩}
```

dove la versione asteriscata impedisce un salto pagina prima della nuova riga; a causa di queste diverse modalità di andare a capo, il comando `\\` è fragile.

Gli ambienti (anche nelle versioni asteriscate) e i comandi che accettano il comando `\\`, a parte il testo normale, sono i seguenti:

\shortstack	\author	\title	\date
eqnarray	*tabbing*	**array**	**tabular**

Gli ambienti *array* e *tabular* sono stati messi in evidenza perché, insieme con i loro parenti stretti definiti mediante i vari pacchetti di estensione descritti in questo testo introduttivo, all'interno delle colonne di tipo testuale, magari avendo dichiarato `\centering`, `\raggedright` o `\raggedleft`, bisogna racchiudere il testo di queste colonne in un gruppo per evitare che il segno di 'a capo' del testo possa essere frainteso per un fine riga della tabella. Si suggerisce di usare al suo posto `\newline` per il testo e `\cr` per il fine riga della tabella, specialmente nell'ultima colonna.

Inoltre se si specifica uno ⟨*spazio*⟩ all'interno di una tabella o di una matrice, se questo è troppo piccolo potrebbe non produrre nessun effetto, perché viene assorbito dalla profondità dello `\strut` che compare in ogni riga delle tabelle e delle matrici.

E.1 La struttura del documento

`\begin{filecontents}{`⟨*nome file*⟩`}`
⟨*contenuto del file*⟩
`\end{filecontents}`
...
`\documentclass[`⟨*lista di opzioni*⟩`]{`⟨*classe*⟩`}`
⟨*preambolo*⟩
`\begin{document}`
⟨*testo del documento*⟩
`\end{document}`

La dichiarazione `\documentclass` può essere preceduta da uno o più ambienti *filecontents* che servono, in particolar modo quando si inviano i file sorgente ad altri collaboratori, per specificare i pacchetti non standard da cui dipende la composizione del documento. Questi file vengono estratti dal file sorgente la prima volta che il documento viene elaborato nella nuova situazione e memorizzati con il loro ⟨*nome file*⟩ nella cartella corrente, cosicché quando la compilazione comincia per davvero, LaTeX dispone di tutti i file di cui ha bisogno, compresi, appunto, i file non standard allegati al file sorgente.

E.2 Periodi e capoversi

E.2.1 Periodi

Apostrofi e virgolette

`'` apostrofo; `'`*testo*`'` virgolette semplici rialzate; `''`*testo*`''` virgolette doppie rialzate; `"<`*testo*`">` virgolette caporali.

Lineati

`-` lineetta; `--` lineato medio per intervalli numerici; `---` lineato lungo per gli incisi e per i dialoghi.

Spazi

`\,` spazio fine; `\␣` spazio interparola; `~` spazio interparola che impedisce di andare a capo; `\@` spazio di fine periodo; inserito dopo una lettera maiuscola e prima di punto fermo serve per evitare che questo sia scambiato per un punto di abbreviazione.

`\frenchspacing` serve per rendere uguali gli spazi dopo qualsiasi segno di interpunzione.

`\nonfrenchspacing` serve per aumentare gli spazi dopo i punti di interpunzione che marcano delle pause maggiori, in particolare dopo i punti fermi. Questa è la situazione di default; lo stile francese viene usato automaticamente durante la composizione della bibliografia; secondo alcuni stimati book designer, la spaziatura alla francese sarebbe da evitare sempre; secondo altri dovrebbe essere l'unica da usare.

Segni speciali

$	`\$`	%	`\%`	{	`\{`	_	`\_`
&	`\%`	#	`\#`	}	`\}`		

Loghi

`\LaTeX` Logo di LaTeX `\TeX` Logo di TeX.

`\today` produce la data del giorno

`\emph{`⟨*testo*⟩`}` Evidenzia ⟨*testo*⟩ scrivendolo in corsivo all'interno di un contesto in tondo e in tondo all'interno di un contesto corsivo.

`\mbox{`⟨*testo*⟩`}` scrive ⟨*testo*⟩ all'interno di una scatola orizzontale in modo che non possa, per esempio, essere divisa in sillabe in fin di riga; questo comando può essere usato anche in un contesto matematico, ma di fatto, se non ci sono motivi speciali, in matematica si possono usare sia i comandi di scelta dei font da usare in un contesto testuale, per esempio `\textrm`, rinunciando però alla prerogativa delle matematica di scegliere il font della giusta dimensione, oppure i comandi `\text` oppure `\intertext` messi a disposizione dal pacchetto **amsmath**.

E.2.2 Capoversi

Un capoverso si può iniziare o terminare con uno dei seguenti comandi; in ogni caso una o più righe assolutamente bianche, senza neanche i segni di commento, serve per terminare un capoverso.

`\noindent` comincia un capoverso senza inserirne il tipico rientro.

`\indent` comincia un capoverso con il tipico rientro, anche quando normalmente esso non sarebbe inserito.

`\par` termina un capoverso e non è più necessario lasciare altre righe bianche.

I capoversi vengono composti secondo le seguenti giustezze:

`\textwidth` è la giustezza normale fissata dal file di classe; se la si vuole cambiare bisognerebbe farlo solo nel preambolo.

`\linewidth` è la giustezza corrente; può essere inferiore o maggiore di `\textwidth` a seconda dell'ambiente all'interno del quale si sta componendo.

`\columnwidth` è la giustezza di una colonna, quando si sta componendo su più colonne.

`\columnsep` è lo spazio di separazione fra due colonne adiacenti.

`\columnseprule` è lo spessore del filetto verticale che separa le colonne quando si compone su due o più colonne.

`\parindent` è l'ammontare del rientro del capoverso.

`\baselineskip` è l'avanzamento di riga stabilito con la scelta del corpo del font che si sta usando, eventualmente modificato dal comando seguente.

`\linespread{`⟨*fattore*⟩`}` modifica l'avanzamento di riga mediante il coefficiente moltiplicativo ⟨*fattore*⟩.

`\parskip` rappresenta una interlinea supplementare da inserire prima di un nuovo capoverso; generalmente è nullo; talvolta è rappresentato da una lunghezza elastica dotata di un valore naturale nullo, ma di un piccolo allungamento, talvolta anche di un piccolo accorciamento, che consente di giustificare le pagine affacciate anche se contengono un numero di righe leggermente diverso. Questo normalmente succede quando, per esempio una formula di una certa dimensione verticale o un nuovo paragrafo dovrebbero cominciare verso la fine di una pagina, ma non c'è abbastanza posto e quindi l'oggetto deve essere spostato alla pagina successiva.

`\enlargethispage{`⟨*lunghezza*⟩`}` allunga la pagina della ⟨*lunghezza*⟩ specificata; per ovvi motivi conviene che questa lunghezza si specificata con un numero intero e piccolo (1 o 2) di righe di testo che, in fondo alla pagina, richiedono un numero intero di avanzamenti di riga. Il comando `\enlargethispage*`, prima di allungare la pagina, cerca di comprimere i contrografismi verticali presenti nella pagina stessa.

E.2.3 Note in calce

`\footnote[`⟨*numero*⟩`]{`⟨*testo*⟩`}`

Se viene specificato il ⟨*numero*⟩ non vi si può fare riferimento in modo simbolico attraverso i comandi `\ref`, `\label` eccetera. Il contatore `footnote` viene rappresentato con un numero a esponente, tranne che nelle note interne all'ambiente *minipage* dove le note sono numerate con lettere minuscole, sempre a esponente. In certe circostanze è possibile separare il comando che inserisce il richiamo della nota dal comando che effettivamente la scrive:

`\footnotemark[`⟨*numero*⟩`]`
...
`\footnotetext[`⟨*numero*⟩`]{`⟨*testo*⟩`}`

Per i parametri stilistici di composizione si veda il prossimo paragrafo.

E.2.4 Note marginali

`\marginpar[`⟨*nota a sinistra*⟩`]{`⟨*nota a destra*⟩`}`

La ⟨*nota a sinistra*⟩ serve facoltativamente per specificare un testo con una modalità di composizione diverso da quello della ⟨*nota a destra*⟩.

I parametri stilistici con cui vengono composte le note in calce e quelle marginali sono le seguenti.

`\footnotesep` è una lunghezza che rappresenta l'altezza dello `\strut` messo all'inizio di ogni nota; aumentando questa altezza ogni nota viene staccata di più dalla precedente.

`\fotnoterule` è il comando con il quale viene composto lo spazio e il filetto orizzontale che separa le note dal testo; deve avere altezza complessiva nulla, quindi bisogna usare anche spazio 'negativo' per compensare lo spessore del filetto di separazione. si può ridefinire questo comando con `\renewcommand`.

`\marginparsep` è la larghezza dello spazio che separa le note marginali dal testo a cui sono affiancate.

`\marginparpush` è la distanza minima fra due note marginali consecutive.

`\marginparwidth` è la giustezza con cui sono composte le note marginali; va da sé che il margine che le deve contenere deve essere sufficientemente largo e deve tenere conto anche di `\marginparsep` e di un buono spazio verso il taglio della pagina affinché dopo il taglio non si debba scoprire che anche parte della nota è stata tagliata.

`\reversemarginpar` è una istruzione booleana che imposta lo scambio del margine nel quale vengono composte le note marginali; agisce solo quando si compone ad una sola colonna, perché a due colonne le note marginali della colonna di sinistra vengono composte comunque a sinistra, e quelle relative alla colonna di destra a destra di questa.

`\normalmarginpar` ristabilisce la composizione delle note marginali nel margine esterno.

E.2.5 Accenti e simboli speciali

Premesso che con l'uso del pacchetto **inputenc** e della tastiera nazionale la necessità di ricorrere ai comandi per gli accenti è fortemente diminuita, tuttavia è utile saper gestire anche i comandi più insoliti al fine di comporre correttamente anche parole in lingue straniere che fanno uso di segni che non compaiono sulla tastiera nazionale.

I comandi per gli accenti testuali sono raccolti nella tabella E.1, mentre quelli per il contesto matematico sono raccolti nella tabella 9.7.

\‘{a}	à	\’{a}	á	\={a}	ā
\v{a}	ă	\^{a}	â	\"{a}	ä
\~{a}	ã	\u{a}	ă	\.{a}	ȧ
\H{a}	ő	\t{oo}	o͡o	\c{t}	ţ
\d{a}	ạ	\b{a}	a̲	\r{u}	ů
\i	ı	\j	ȷ	\oe	œ
\OE	Œ	\ae	æ	\AE	Æ
\aa	å	\AA	Å	\o	ø
\O	Ø	\l	ł	\L	Ł
\ss	ß	?‘	¿	!‘	¡
\dag	†	\ddag	‡	\S	§
\P	¶	\copyright	©	\pounds	£

Tabella E.1: I comandi per gli accenti testuali e i simboli speciali di molte lingue straniere

E.3 Suddivisione del testo e indici

E.3.1 Comandi di sezionamento

Tutti i comandi di sezionamento propriamente detti, e gli altri comandi simili hanno una sintassi comune:

comando[⟨*titolo breve*⟩]{⟨*titolo lungo*⟩}

Essi sono:

```
\part            \chapter     \section        \subsection
\subsubsection   \paragraph   \subparagraph   \caption
```

Il comando `\chapter` non è definito per la classe `article`; nessun comando di sezionamento è definito per la classe `letter`. Il livello di sezionamento `\part` non è obbligatorio; tutti gli altri comandi devono essere contenuti sotto un comando del livello precedente.

Le parole che eventualmente vengono scritte nelle intestazioni, coma ‘Part’ o ‘Parte’, ‘Chapter’ o ‘Chapître’, vengono impostate tramite l’opportuna chiamata al pacchetto **babel** con le opzioni per la lingua del documento. In ogni caso esse sono contenute in comandi del tipo `\partname`, `\chaptername`, `\figurename`, `\tablename` eccetera, per cui possono venire ridefinite a piacere.

Ogni comando di sezionamento oltre a comporre l’intestazione della divisione specifica mediante il ⟨*titolo lungo*⟩, scrive negli opportuni file accessori il ⟨*titolo breve*⟩ da inserire negli opportuni indici.

Ogni divisione del documento può venire numerata e può comparire nell’indice; questi due fatti sono controllati dai valori numerici contenuti nei due contatori `secnumdepth` e `tocdepth`; Le sezioni sono numerate se il loro livello è associato ad un numero minore o uguale a quello contenuto in `secnumdepth`; i loro titoli vanno a finire nell’indice se il loro livello corrisponde ad un numero minore o uguale a quello contenuto in `tocdepth`; si veda la tabella E.2. Per produrre gli indici si usano i comandi:

```
\tableofcontents    \listoffigures    \listoftables
```

Sezionamento	Livello
`\part`	−1
`\chapter`	0
`\section`	1
`\subsection`	2
`\subsubsection`	3
`\paragraph`	4
`\subparagraph`	5
`\caption`	—

Tabella E.2: Numeri associati a livello di sezionamento

Le informazioni per produrre questi indici sono contenute in tre file ausiliari le cui estensioni sono rispettivamente `.toc`, `.lof` e `.lot`.

Le versioni asteriscate di questi comandi producono solo l'intestazione ma non producono niente nell'indice. Il comando `\caption` non possiede la variante asteriscata. Sebbene le versioni asteriscate non producano nessuna voce nei file indice, tuttavia le si può far comparire se si procede a mano inserendo dopo i comandi espliciti asteriscati il comando `\addcontentsline` con la seguente sintassi:

`\addcontentsline{`⟨*estensione*⟩`}{`⟨*livello*⟩`}{`⟨*voce*⟩`}`

dove ⟨*estensione*⟩ è l'estensione del file ausiliario per l'indice specifico, ⟨*livello*⟩ è il nome (non il numero) del livello di sezionamento, espresso dallo stesso nome del comando, senza però il backslash; La ⟨*voce*⟩ è quanto viene scritto nel file indice. Generalmente essa è costituita dall'espressione

`\protect\numberline{`⟨*numero*⟩`}{`⟨*titolo*⟩`}`

dove ⟨*numero*⟩ è il numero della sezione, per esempio 3.8 per l'ottavo paragrafo del capitolo tre; va da sé che una sezione non numerata ha questo campo vuoto. Il ⟨*titolo*⟩ non è altro che il titolo (breve) della sezione.

Per l'indice analitico e il glossario, i vari elementi vengono raccolti mediante i rispettivi comandi `\index` e `\glossary`, ma il contenuto del loro argomento varia molto a seconda di come si intenda usare l'informazione raccolta. In ogni caso nessuna informazione viene raccolta se nel preambolo mancano i comandi `\makeindex` e, rispettivamente, `\makeglossary`; l'indice analitico e/o il glossario non vengono composti se non si da esplicitamente il comando di comporli, di solito attraverso comandi messi a disposizione da pacchetti esterni. Altrimenti bisogna usare gli ambienti *index* e *glossary* specificando a mano il nome del file che contiene l'indice o il glossario elaborati adeguatamente.

E.4 Classi, pacchetti e stili delle pagine

E.4.1 Classe del documento

La scelta della classe va eseguita con il comando:

`\documentclass[`⟨*opzioni*⟩`]{`⟨*classe*⟩`}`

Le classi standard sono `book`, `report`, `article`, `letter`, `slides`, `minimal`, `proc`, `ltxdoc` insieme alle numerosissime altre classi non standard che popolano gli archivi internazionali.

Le opzioni standard sono le seguenti

10pt, definisce il corpo di default per il documento; però può venire trasmessa anche ad altri pacchetti, fra i quali **type1ec** il quale con questa opzione ricava i disegni di tutti i caratteri dai font in corpo 10 pt, ingrandendoli o rimpicciolendoli a seconda delle richieste del compositore. Questo modo di procedere è accettabile solo quando si sta lavorando con font vettoriali. Il file di uscita risulta meno ingombrante, ma la qualità peggiora un poco, ma visibile ad occhio nudo. Tra l'altro si noti la differenza di resa dei font ingranditi o rimpiccioliti; qui si mostra lo stesso testo riportato al corpo 10 partendo da un font di corpo 5 e da un font di corpo 17: partendo da un font di corpo 5 e partendo da un font di corpo 17. L'effetto opposto si otterrebbe partendo da un font di corpo 10 riducendolo al corpo 5 o ingrandendolo al corpo 17.

11pt, *12pt*, servono per scegliere il corpo da 11 pt o da 12 pt come corpo normale.

letterpaper, *legalpaper*, *executivepaper*, *a4paper*, *a5paper*, *b5paper*, servono per scegliere il supporto della carta secondo i formati specificati con le sigle suddette, precisamente

letter	8,5 in × 11 in	A4	210 mm × 297 mm
legal	8,5 in × 14 in	A5	148 mm × 210 mm
executive	7,25 in × 10,5 in	B5	176 mm × 250 mm

Siccome l'opzione di default è *letterpaper* per scrivere in Europa è sempre necessario specificare un formato ISO, quasi sempre *a4paper*.

Le dimensioni così specificate vengono passate anche al file `.pdf` nel caso che si usi pdfLaTeX, in quanto quel formato ha bisogno di conoscere le dimensioni del supporto di uscita. Il file `.dvi` a stretto rigore non ne ha bisogno, ma poi diventa necessario se questo tipo di file viene convertito nei file `.ps` oppure `.pdf`.

landscape, serve per scambiare fra di loro la larghezza e l'altezza del supporto.

final, *draft*, servono per specificare se si sta componendo la versione finale, oppure una bozza; quando si compone una bozza, vengono evidenziate le righe fuori giustezza con un vistoso rettangolo nero e non vengono importate le figure, ma al loro posto viene segnato solo un rettangolo delle giuste proporzioni con il nome del file grafico scritto dentro.

oneside, *twoside*, servono per specificare se si vuole comporre solo sul recto o anche sul verso della carta.

openright, *openany*, servono per specificare sei i capitoli possono iniziare solo sulle pagine di destra oppure su qualunque pagina.

onecolumn, *twocolumn*, servono per specificare si si vuol comporre su un'unica colonna o su due colonne.

notitlepage, *titlepage*, servono per specificare se il titolo del documento va in testa alla prima pagina, nella quale comincia subito anche il testo, oppure se si deve riservare una pagina solo per il titolo.

openbib, serve per comporre la bibliografia (nelle classi che la riconoscono) con uno stile più aperto. Le righe oltre la prima risultano con un rientro maggiore specificato mediante `\bibindent`.

leqno, serve per specificare che si desidera numerare le equazioni con il numero collocato a filo del margine sinistro della gabbia del testo.

fleqn, serve per specificare che non si desiderano le equazioni centrate, ma rientrate di una quantità fissa dal margine sinistro; questa quantità viene specificata mediante `\mathindent` che di default vale 50 pt.

E.4.2 Pacchetti

La scelta del pacchetto si esegue con il comando `\usepackage` con la sintassi

`\usepackage[`⟨*opzioni*⟩`]{`⟨*pacchetto*⟩`}[`⟨*data*⟩`]`

dove ⟨*data*⟩ è una data in forma ISO (*aaaa/mm/gg*) che rappresenta il limite inferiore della data corrispondente alla versione che si vuole usare; nei circa 15 anni da quando esiste LaTeX 2ε i vari pacchetti sono stati aggiornati così che versioni successive non solo hanno (presumibilmente) meno errori o 'features' degli stessi pacchetti nelle loro versioni precedenti, ma anche funzionalità maggiori. Ecco che specificando la data si può essere sicuri di usare una versione abbastanza recente che presenta le funzionalità desiderate. I pacchetti standard sono:

alltt serve per definire l'ambiente *alltt* dove quasi tutti i caratteri speciali perdono il loro significato specifico e vengono usati come caratteri 'normali'.

amsmath costituisce la principale estensione per gli ambienti matematici.

babel consente di comporre in diverse lingue; attenzione; **babel** sceglie le specifiche della lingua, compresa la sillabazione, ma non è **babel** che carica la sillabazione; vedi l'appendice D.

color serve per gestire i colori dei font e degli sfondi.

graphics serve per gestire le manipolazioni grafiche. Benché sia il pacchetto che svolge il lavoro richiesto, per l'utente è più comodo riferirsi al pacchetto seguente.

graphicx gestisce le operazioni grafiche e gli effetti speciali con una interfaccia utente più efficace; in realtà è **graphicx** che si preoccupa di caricare **graphics** e di passargli i comandi tradotti nella sua sintassi specifica.

ifthen permette di eseguire un fine controllo logico con una interfaccia utente più semplice di quella che offrono i comandi primitivi di TeX.

latexsym consente di usare i simboli speciali di LaTeX 209; questo pacchetto è più per compatibilità con il passato che altro; tutti quei simboli sono accessibili anche con il pacchetto **amsmath** o il suo subalterno **amsfonts**.

makeidx serve per gestire comodamente la creazione dell'indice analitico.

pict2e estende le capacità grafiche dell'ambiente *picture* di LaTeX. Queste estensioni sono disponibili solo dal 2003, e il pacchetto ha raggiunto una situazione abbastanza stabile solo dal 2004; si vede dunque come l'opzione della ⟨*data*⟩ non sia una cosa tanto insolita da usare.

showidx consente di scrivere le voci raccolte con `\index` direttamente sulla pagina dove sono state trovate; durante la lavorazione di un documento questa opzione si rivela molto utile per selezionare e confrontare le varie voci, sia per ridurre i duplicati sia per scegliere i riferimenti più adatti.

E.4.3 Stili delle pagine

Ogni classe inizializza lo stile delle pagine normali secondo le prescrizioni stilistiche generali della classe stessa. Il comando per scegliere lo stile è

```
\pagestyle{⟨stile⟩}
```

Il comando

```
\thispagestyle{⟨stile⟩}
```

imposta lo stile della pagina solo per la pagina corrente.

Gli stili predefiniti nelle classi standard sono i seguenti.

`plain` contiene solo il piedino con il numero della pagina.

`empty` sia il piedino sia la testatina sono vuoti.

`headings` il piedino è vuoto ma la testatina è elaborata e automaticamente contiene sul verso delle pagine scritte con l'opzione *twoside* il titolo (breve) del capitolo e nella testatina del recto il titolo (breve) del paragrafo attivo all'inizio della pagina.

Questi titoli sono forniti dai comandi di sezionamento in modo automatico; se si desiderano testatine specifiche, che contengano elementi diversi o titolini correnti diversi bisogna usare i comandi `\markright` e `\markboth`, oppure bisogna servirsi di pacchetti di estensione come, per esempio, **fancyhead**. Oppure bisogna servirsi di classi diverse da quelle standard.

`myheadings` in senso generale lo stile è simile allo stile `headings`, salvo che le informazioni da inserire nelle testatine debbono essere fornite mediante i comandi `\markright` e `\markboth`.

La sintassi di questi comandi è la seguente:

```
\markright{⟨titolino di destra⟩}
\markboth{⟨titolino di sinistra⟩}{⟨titolino di destra⟩}
```

La numerazione delle pagine viene specificata con il comando

```
\pagenumbering{⟨stile di numerazione⟩}
```

dove gli stili di numerazione disponibili sono i seguenti:

`arabic` serve per la normale numerazione con cifre arabe.

`roman` serve per la numerazione romana in lettere minuscole.

`Roman` serve per la numerazione romana in lettere maiuscole

`alph` serve per la numerazione alfabetica in lettere minuscole.

`Alph` serve per la numerazione alfabetica in lettere maiuscole.

`fnsymbol` serve per rappresentare i numeri con la stessa sequenza di simboli usata per le note, per esempio l'asterisco, la spada, la doppia spada e via di seguito. I simboli in totale sono nove, quindi non è possibile numerare più di 10 cose. Generalmente questo tipo di numerazione non viene usato per le pagine, ma lo si usa abbastanza spesso per le note, in particolare per quelle del frontespizio.

Difficilmente si useranno numerazioni alfabetiche perché il numero totale delle pagine da numerare potrebbe superare 26, ma per le pagine le numerazioni arabe e romane vengono usate spesso.

Lo stile della pagina è vistosamente differente se si compone su una sola colonna o su due o più colonne; le classi standard consentono al massimo due colonne, ma mediante il pacchetto **multicol** non è difficile aumentare il numero delle colonne.

Quando però si usano le classi standard e si specifica di comporre ad una sola colonna, è possibile comporre alcune pagine su due colonne, consentendo di comporre anche una intestazione ad una colonna. al contrario se si è specificata l'opzione per comporre su due colonne, allora è possibile comporre parte delle pagine su una sola colonna. I comandi da usar sono:

```
\twocolumn[⟨intestazione ad una colonna⟩]
\onecolumn
```

Entrambi i comandi iniziano sempre una nuova pagina poi cominciano a comporre come dice il loro nome.

Ovviamente lo stile della pagina dipende anche dalla giustezza e dalla sua relazione con il formato della carta, quindi dei margini. Le giustezze orizzontali e verticali sono specificate mediante le lunghezze `\textwidth` e `\textheight` e i margini sono specificati mediante le lunghezze `\oddsidemargin` per le pagine di destra, `\evensidemargin` per le pagine di sinistra, `\topmargin`; i margini laterali sono comunque i margini sinistri, e le parole 'odd' e 'even' (dispari o pari) si riferiscono alla posizione della pagina in relazione alla sua numerazione, sapendo che il verso di tutte le pagine è sempre pari e il recto sempre dispari. Questi margini sono riferiti allo spigolo superiore sinistro del foglio di carta e tutti sono 'un pollice' di meno di quello che ci si aspetterebbe; questo dipende dal fatto che inizialmente era previsto che tutti i driver di uscita avrebbero inserito di default un pollice di margine.

Non è il caso di mettersi a giocare con questi margini; se proprio si deve modificare l'aspetto geometrico di una pagina, è meglio usare una classe che consenta di farlo, oppure il pacchetto **geometry**.

E.4.4 Il frontespizio

È vero che se uno deve comporre un frontespizio in modo decoroso, è opportuno che studi il layout e la scelta del corpo e dello stile dei font in modo professionale. Tuttavia nei rapporti e negli articoli anche la modesta composizione di default operata dalla classi standard di LaTeX può essere accettabile.

Bisogna ovviamente specificare l'opzione *titlepage*; ma bisogna anche specificare nel preambolo un certo numero di informazioni.

`\title{`⟨*titolo*⟩`}` serve per specificare il titolo del documento; se questo titolo è sufficientemente lungo lo si può comporre su più righe inserendo il comando `\\`. Bisogna però ricordarsi di non andare a capo fra un articolo e il nome, fra una preposizione e il suo complemento, fra avverbi brevi, come 'non', e quanto segue; fa parte dell'eleganza della composizione. Riassumendo: nei titoli non si deve mai andare a capo dopo le congiunzioni, gli articoli, le preposizioni e i brevi avverbi; per evitarlo si deve usare il comando di legatura costituita dalla tilde `~`.

`\author{`⟨*nomi*⟩`}` serve per specificare i nome degli autori; ma se gli autori sono più di uno bisogna separarli con `\and`; si può andare a capo usando `\\`. Attenzione: per ciascun autore sempre il o i nomi propri prima del o dei cognomi!

`\date{`⟨*testo*⟩`}` il testo di questo comando potrebbe essere una data, come suggerisce il nome del comando, ma potrebbe anche essere il nome di una conferenza, un luogo, qualunque informazione che permetta di identificare il documento oltre al suo titolo e ai suoi autori (i quali potrebbero aver prodotto un articolo con lo stesso titolo ad un altro convegno...).

`\thanks{`⟨*testo*⟩`}` questo comando può essere appeso sia ai nomi degli autori, per esempio per specificarne l'afferenza, sia al titolo, per esempio per associargli il nome dell'ente finanziatore delle ricerca, sia al testo specificato con `\date`. Esso funziona come il comando per inserire delle note a piè di pagina, solo che il riferimento di queste note non è né un numero né una lettera, ma un generico simbolo tratto da una lista che contiene l'asterisco, la spada, la spada doppia, eccetera.

Il tutto viene eseguito dando il comando `\maketitle`. Alternativamente e con risultati esteticamente variabili a seconda del gusto del compositore, si può comporre il frontespizio aprendo l'ambiente *titlepage* e scrivendoci dentro quello che si desidera, dove lo si desidera, con i font che si desiderano; talvolta la cosa produce buoni risultati, talaltra è meglio affidarsi alla composizione di default.

Nei rapporti e negli articoli spesso il frontespizio può contenere anche un breve riassunto; in LaTeX questo viene composto mediante l'ambiente *abstract*. Il riassunto può comparire come prima cosa dell'articolo e se questo è composto su due colonne, il sunto si trova nella prima colonna, subito sotto al titolo e alle informazioni composte con `\maketitle`.

E.5 Testi in display

E.5.1 Citazioni e poemi

Gli ambienti per i testi in display sono:

```
\begin{quote} ⟨testo⟩ \end{quote}
\begin{quotation} ⟨testo⟩ \end{quotation}
\begin{verse} ⟨testo⟩ \end{verse}
```

E.5.2 Liste

Gli ambienti predefiniti per le liste sono:

```
\begin{itemize}
\item[⟨contrassegno⟩] ⟨testo⟩
...
\end{itemize}
```

```
\begin{enumerate}
\item[⟨contrassegno⟩] ⟨testo⟩
...
\end{enumerate}
```

```
\begin{description}
\item[⟨contrassegno⟩] ⟨testo⟩
...
\end{description}
```

Per i primi due ambienti il ⟨*contrassegno*⟩ è davvero facoltativo; per l'ambiente *description*, benché appaia come argomento facoltativo, in realtà esso è obbligatorio, perché non avrebbe senso dare una descrizione di nulla. Tuttavia se non si specifica nulla, nemmeno le parentesi quadre, si comincia una nuova descrizione senza etichetta che può essere vista come un nuovo capoverso. Non sembra una buona idea sfruttare così male le potenzialità di LaTeX.

L'ambiente principale, con il quale sono costruite tutte le liste, però, è l'ambiente *list*, che è completamente configurabile in ogni dettaglio. La sintassi è:

```
\begin{list}{⟨contrassegno di default⟩}{⟨dichiarazioni⟩}
\item[⟨contrassegno personalizzato⟩] ⟨testo⟩
...
\end{list}
```

Il ⟨*contrassegno di default*⟩ è il modo di comporre il contrassegno che *list* produce quando non viene esplicitato un ⟨*contrassegno personalizzato*⟩ con il comando `\item`. Le ⟨*dichiarazioni*⟩, invece, sono una serie di istruzioni che specificano il modo di comporre eseguito da *list*. Ci sono numerose possibilità che è meglio descrivere una alla volta.

`\topsep` è la distanza che separa il testo precedente alla lista dalla prima voce della lista. La stessa distanza viene posta dopo la chiusura della lista.

`\partopsep` è lo spazio aggiuntivo aggiunto prima e dopo la lista se questa comincia un nuovo capoverso, cioè se nel file sorgente una riga completamente bianca precede il comando di apertura dell'ambiente.

`\itemsep` è lo spazio aggiuntivo che viene messo prima di una voce della lista se questa è preceduta da una riga bianca.

`\parsep` Siccome dentro le liste i capoversi solitamente non vengono rientrati, allora si può usare un piccolo spazio aggiuntivo fra due capoversi appartenenti alla stessa voce; questo spazio di separazione fra i *paragraphs* si chiama appunto `\parsep`.

`\leftmargin` è la rientranza di ogni voce della lista rispetto al testo circostante; liste annidate hanno rientranze sempre maggiori; il file di classe specifica le varie successive rientranze mediante i comandi `\leftmargini`, `\leftmarginii`, `\leftmarginiii` e `\leftmarginiv`.

`\rightmargin` è la distanza orizzontale fra il margine destro della lista corrente e il margine destro del testo che contiene la lista.

`\listparindent` è il rientro extra aggiunto (o anche tolto se ha un valore negativo) alla rientranza di ogni linea di ciascuna voce, tranne la prima riga; tutte le classi standard hanno questo parametro impostato a zero, ma non è impossibile che in altre classi o in liste personalizzate non si possa usare questo parametro.

`\itemindent` è la rientranza della prima riga di ogni voce della lista. Può anche avere un valore negativo.

`\labelsep` è la distanza minima che separa il ⟨*contrassegno*⟩ dal resto del testo nella prima riga di ogni voce.

`\labelwidth` è la larghezza prevista per il ⟨*contrassegno*⟩; a seconda della personalizzazione se questo contrassegno fosse più largo o più stretto di questa larghezza, il contrassegno viene fatto sporgere a sinistra, oppure viene fatto sporgere a destra ma cominciando il testo della voce corrispondentemente più a destra.

`\makelabel{`⟨*contrassegno*⟩`}` è la macro che effettivamente compone il contrassegno.

`\usecounter{`⟨*contatore*⟩`}` specifica che il ⟨*contatore*⟩ è quello usato per numerare gli elementi della lista e lo rende usabile con i comandi `\label` e `\ref`.

E.5.3 Testo composto verbatim

Si possono usare due comandi e due ambienti per comporre del testo in modo verbatim; questo modo di comporre è quello che serve quando bisogna esporre, per esempio, dei brani scritti in un linguaggio di programmazione nel quale si fa uso dei caratteri speciali di TEX (ovviamente con altri significati). I comandi sono

`\verb`*simbolo*⟨*testo da riprodurre*⟩*simbolo*
oppure
`\verb*`*simbolo*⟨*testo da riprodurre*⟩*simbolo*

Il *simbolo* funziona da delimitatore del ⟨*testo da riprodurre*⟩; la versione senza asterisco riproduce gli spazi come spazi; la versione con l'asterisco riproduce gli spazi con il segno speciale ␣. Il ⟨*testo da riprodurre*⟩ deve comparire in una sola riga nel file sorgente.

Gli ambienti sono invece:

```
\begin{verbatim}⟨testo da riprodurre verbatim⟩\end{verbatim}
\begin{verbatim*}⟨testo da riprodurre verbatim⟩\end{verbatim*}
```

Il ⟨*testo da riprodurre verbatim*⟩ si può svolgere su diverse righe, anzi di solito è composto di diverse righe. L'unica riga che non si può riprodurre è `\end{verbatim}` con o senza asterisco. L'ambiente asteriscato, come il comando asteriscato, riproduce lo spazio in modo visibile con il carattere ␣.

Il comando `\verb` è fragilissimo e non può apparire come argomento di nessun altro comando (anche nella versione asteriscata).

Il pacchetto standard **alltt** definisce un ambiente *alltt* da usare come l'ambiente *verbatim* dove tutti i caratteri sono diventati caratteri 'normali, salvo `\`, `{` e `}`.

E.6 Formule matematiche

E.6.1 Formule

La matematica in linea può essere delimitata dai seguenti delimitatori:

```
$ ⟨formula in linea⟩ $
\( ⟨formula in linea⟩ \)
\begin{math} ⟨formula in linea⟩ \end{math}
```

Si consiglia di non usare il primo metodo, anche se è più comodo da scrivere, perché si perde la diagnostica di LaTeX nel caso che ci si dimentichi di uno dei due delimitatori. L'ambiente *math*, come si può ben capire, non viene usato molto spesso. I delimitatori `\(` e `\)` sono i migliori, ma sono fragili. Invece i delimitatori `$`...`$` sono robusti.

Il comando

```
\ensuremath{⟨formula⟩}
```

permette di comporre una ⟨*formula*⟩ in linea garantendo che essa venga composta correttamente anche se il comando viene emesso in modo testuale. È comodo per definire macro che debbono potersi usare sia nel modo testo sia nel modo matematico.

Usando il pacchetto **babel** è disponibile il comando

```
\textormath⟨per il modo testo⟩⟨per il modo matematico⟩
```

che produce un effetto analogo a quello di `\ensuremath`, con una notevole differenza; usato per definire un comando da usare sia in modo testo sia in modo matematico, può eseguire composizioni diverse nell'uno o nell'altro modo senza che il compositore debba preoccuparsi di verificare il modo di composizione; `\ensuremath`, invece, compone il suo argomento sempre in modo matematico,

all'occorrenza entrando ed uscendo da questo modo, al fine di lasciare TEX nello stesso stato nel quale si trovava prima dell'esecuzione del comando.

Per la matematica in display si hanno gli ambienti

```
\[ ⟨formula in display⟩ \]
\begin{displaymath} ⟨formula in display⟩ \end{displaymath}
\begin{equation} ⟨formula in display⟩ \end{equation}
```

I primi due ambienti non assegnano un numero alla ⟨*formula in display*⟩, mentre il terzo ambiente assegna un numero a detta formula. L'ambiente *displaymath* viene usato raramente visto che quello definito con `\[...\]` è molto più comodo da usare. L'ambiente *equation* non ha sostituti.

LATEX offrirebbe anche gli ambienti *eqnarray* e *eqnarray** ma, per i motivi già esposti nel capitolo 10, è preferibile non servirsene; al contrario per comporre formule e sistemi di formule in display è molto meglio servirsi del pacchetto **amsmath**.

Dentro agli ambienti *equation* e *eqnarray* (all'interno di ciascuna riga) è possibile inserire il comando `\label` così da definire una etichetta mnemonica per richiamare il numero della formula con `\ref` e/o con `\pageref`. Dentro *eqnarray* si può usare `\nonumber` per evitare che LATEX assegni un numero ad una espressione che non si vuole numerare. Questo ambiente consentirebbe anche di spezzare una singola lunga espressione matematica su più righe, ricorrendo anche al comando suddetto e a `\lefteqn`; ma per questo scopo è meglio servirsi degli opportuni ambienti predisposti allo scopo dal pacchetto **amsmath**.

Le espressioni in display usano un certo numero di parametri per comporre secondo diversi stili.

`\jot` serve per aggiungere altro spazio fra una equazione e l'altra in un sistema di equazioni.

`\mathindent` serve per fissare la rientranza sinistra delle equazioni quando queste sono composte allineate a sinistra se si usa l'opzione *fleqn* per la classe del documento.

`\abovedisplayskip` serve per definire il contrografismo che precede una equazione in display; viene usato un contrografismo più piccolo quando la riga prima del display è molto corta e l'espressione matematica potrebbe dare l'impressione ottica di essere preceduta da uno spazio troppo grande.

`\abovedisplayshortskip` è appunto questo contrografismo superiore più piccolo.

`\belowdisplayskip` è il contrografismo da inserire dopo una formula per distanziarla dal testo seguente.

`\belowdisplayshortskip` serve per inserire un contrografismo più piccolo se l'espressione matematica è seguita da una riga di testo molto corta.

Va da sé che i `...shortskip` non vengono usati quando si compone con l'opzione *fleqn* perché le espressioni sono tutte allineate a sinistra, a meno del loro rientro non molto importante (una cinquantina di punti, circa 15 mm).

Per comporre le espressioni matematiche si usano spesso piccoli comandi o piccole strutture di cui è facile dimenticarsi.

Esponenti gli esponenti si inseriscono mediante il comando `^`

```
^{⟨esponente⟩}
```

Con l'opzione *italian* di **babel** è disponibile il comando `\ap{⟨apice⟩}` per inserire un apice in tondo quando si è in modo matematico; e un apice con il font corrente quando si è in modo testo.

Apici L'apice ′ o il doppio apice ″ si inseriscono con uno o due apostrofi, che in modo matematico vengono composti come apici.

Pedici e deponenti I pedici vengono inseriti con il comando `_`

```
_{⟨pedice⟩}
```

Con l'opzione *italian* di **babel** è possibile usare il comando `\ped{⟨pedice⟩}` per inserire un pedice in tondo in modo matematico e per inserire un deponente con il font corrente in modo testo.

Frazioni Le frazioni vengono composte con il comando

```
\frac{⟨numeratore⟩}{⟨denominatore⟩}
```

Bisogna ricordare che questo comando compone le frazioni in display come ci si aspetterebbe, ma le compone in modo testo sia nelle espressioni matematiche in linea, sia nelle sottoespressioni come per esempio un altro numeratore o denominatore; in questi casi è preferibile servirsi delle frazioni a barra obliqua, piuttosto che di quelle a barra orizzontale.

Radici le radici quadrate o con altro ⟨*indice*⟩ si compongono con

```
\sqrt[⟨indice⟩]{⟨radicando⟩}
```

Se l'⟨*indice*⟩ vale 2 non lo si indica.

Ellissi Le parti omesse (ellissi) vengono sostituite con i soliti tre puntini; in matematica se ne hanno di diversi tipi: `\dots` (…) si può usare sia in modo matematico sia in modo testo e produce i soliti puntini a livello della linea di base; `\ldosts` produce lo stesso effetto, ma è da usare in modo matematico; `\cdots` (⋯) produce i tre puntini allineati con il segno −; `\vdots` (⋮) produce tre puntini verticali che servono per sostituire in verticale gli elementi omessi da colonne di matrici; infine `\ddots` (⋱) produce tre puntini in diagonale, utili per riempire una parte omessa dal cuore di una matrice.

E.6.2 Simboli, accenti, delimitatori e grandi operatori

Tutti i simboli di qualunque genere usabili in matematica sono raccolti nelle tabelle 9.2–9.8; nelle tabelle 10.1–10.2, invece, sono raccolti gli ulteriori simboli disponibili con il pacchetto **amsmath**.

E.6.3 Impilare gli oggetti matematici

Una (breve) espressione può venire sopralineata con

`\overline{`⟨*espressione*⟩`}`

Analogamente si può sottolineare una espressione con

`\underline{`⟨*espressione*⟩`}`

Anche gli accenti matematici possono venire impilati, ma si tratta semplicemente di racchiudere fra le graffe che delimitano l'argomento del primo accento una espressione già accentata. Fra i comandi non presenti nella tabella 9.7 si può annoverare anche `\widetilde`.

Invece `\stackrel` permette di costruire degli operatori di relazione impilando uno sopra l'altro due segni distinti, scelti fra i vari simboli disponibili:

`\stackrel{`⟨*elemento superiore*⟩`}{`⟨*elemento inferiore*⟩`}`

E.6.4 Spaziatura matematica

Non si può raccomandare mai abbastanza l'opportunità di non spaziare le espressioni matematiche. I comandi di spaziatura hanno senso se e solo se la spaziatura di default non è adeguata ai simboli che compaiono uno accanto all'altro in una espressione matematica. Se, quindi, si inserirà una qualche spaziatura, lo si farà solo dopo aver corretto le bozze controllando accuratamente che questa spaziatura sia praticamente impercettibile.

`\,`	spazio sottile	`\:`	spazio medio
`\!`	spazio sottile negativo	`\;`	spazio grande

Il comando `\,` può essere usato anche in modo testo. Naturalmente si potrebbero anche usare `\quad` e `\qquad` oltre ai comandi di spaziatura del modo testo. Tuttavia merita di segnalare i comandi primitivi `\mskip` e `\mkern` che accettano come argomenti (non racchiusi fra nessun tipo di parentesi) degli spazi espressi mediante le unità di misura chiamate `mu`, valide solo in matematica; 18 mu equivalgono a 1 em, ma in matematica l'unità 'em' cambia grandezza a seconda del font in uso, in particolare per il corpo principale della formula, per gli apici ed i pedici di primo ordine e per gli apici ed i pedici di secondo ordine.

E.6.5 Font matematici

Per avere una intera formula composta con caratteri medi o con caratteri neri bisogna specificarlo prima di entrare in modo matematico:

`\boldmath`
⟨*ambiente matematico*⟩
`\unboldmath`

Altrimenti i vari simboli letterali e/o gli operatori possono essere resi con font diversi se si usano i comandi seguenti:

`\mathrm`	tondo	`\mathsf`	senza grazie
`\mathit`	*corsivo matematico*	`\mathtt`	`spaziatura fissa`
`\mathbf`	**nero**	`\mathcal`	$\mathcal{CALLIGRAFICO}$

Si noti che il corsivo matematico e il corsivo per il testo sono due font con proprietà completamente diverse; per esempio il primo non contiene legature, per cui la parola *affine* verrebbe composta $af\,fine$.

E.6.6 Stili di composizione

I quattro stili di composizione della matematica sono

```
\displaystyle
\textstyle
\scriptstyle
\scriptscriptstyle
```

Lo stile `\textstyle` differisce dal `\displaystyle` nel senso che è più raccolto in verticale: gli apici e i pedici sono più vicini all'asse matematico; i limiti superiori e inferiori sono composti accanto all'operatore come se fossero apici o pedici; le frazioni sono composte in `\scriptstyle` per mantenere limitato l'ingombro verticale in modo che non sia necessario spaziare le righe del testo. È per questo che bisogna stare attenti, specialmente con le frazioni, a non renderle troppo piccole; piuttosto che una frazione troppo piccola, è preferibile una barra obliqua, senza rimpicciolire numeratori e denominatori, nemmeno quando sono puramente numerici, come in $^2/_3$; questa pratica è 'deprecata' dalla norma UNI 2950.

E.7 Definizioni, numeri e programmazione

E.7.1 Comandi di definizione

Nuove macro possono essere definite, ridefinite o rese disponibili, se non lo sono già, mediante i comandi:

```
\newcommand{⟨comando⟩}[⟨numero argomenti⟩][⟨default⟩]{⟨definizione⟩}
\renewcommand{⟨comando⟩}[⟨numero argomenti⟩][⟨default⟩]{⟨definizione⟩}
\providecommand{⟨comando⟩}[⟨numero argomenti⟩][⟨default⟩]{⟨definizione⟩}
```

Sono tutti comandi fragili, ma del resto non sembra opportuno usarli negli argomenti di altri comandi. Sono già stati descritti in dettaglio nel capitolo 16. Tutti dispongono della versione asteriscata; la differenza sembra minima, ma è importante per evitare perdite di tempo quando si commettono errori nell'uso dei comandi definiti mediante queste istruzioni. In termini di gergo TeX, i comandi senza asterisco sono `\long`, mentre quelli con asterisco non lo sono. Un 'long command' che accetta uno o più argomenti è in grado di elaborare anche argomenti composti di più capoversi; per esempio `\parbox` è un 'long command'. I comandi che non hanno questo attributo non consentono che i loro argomenti contengano una linea bianca e/o un comando esplicito di fine capoverso, `\par`; se questo succedesse essi arresterebbero subito la composizione con un messaggio di errore. Questo fatto succede abbastanza frequentemente

quando ci si dimentica la parentesi graffa di chiusura di un argomento. Ma con i comandi 'corti' l'errore viene trovato subito con la normale diagnostica di LaTeX, mentre con i comandi lunghi l'errore potrebbe venire segnalato o alla fine del file oppure perché la memoria di TeX è satura.

E.7.2 Comandi per la definizione di ambienti

Gli ambienti sono definiti o ridefiniti mediante i comandi

```
\newenvironment{⟨nome⟩}[⟨numero argomenti⟩][⟨default⟩]%
                    {⟨comandi di apertura⟩}{⟨comandi di chiusura⟩}
\renewenvironment{⟨nome⟩}[⟨numero argomenti⟩][⟨default⟩]%
                    {⟨comandi di apertura⟩}{⟨comandi di chiusura⟩}
```

Si veda il capitolo 16 per maggiori dettagli ed esempi.

E.7.3 Teoremi

LaTeX mette a disposizione dei comandi per definire degli pseudo ambienti al fine di comporre gli enunciati di teoremi o di simili enunciati. Il comando `\newtheorem` consente due forme di definizione:

```
\newtheorem{⟨nome⟩}{⟨titolino⟩}[⟨contatore dominante⟩]
\newtheorem{⟨nome⟩}[⟨numerato come⟩]{⟨titolino⟩}
```

dove:

⟨*nome*⟩ rappresenta il nome dello pseudo ambiente; potrà essere, per esempio, teorema, lemma, definizione, eccetera. Per cui l'enunciato verrà racchiuso all'interno di `\begin{teorema}` ⟨*enunciato*⟩ `\end{teorema}`.

⟨*titolino*⟩ 'la parola che identifica l'enunciato; potrà essere Teorema, Lemma, Definizione, eccetera. Non si confonda il nome dell'ambiente con il titolino dell'enunciato.

⟨*contatore dominante*⟩ è il contatore del capitolo, se si vuole che la numerazione del nuovo enunciato sia iniziata da 1 ad ogni nuovo capitolo; sarà il contatore dei paragrafi, se si desidera che la numerazione ricominci da 1 ad ogni nuovo paragrafo; eccetera.

⟨*numerato come*⟩ è invece il nome dello pseudo ambiente del quale si vuole condividere la numerazione; Questo deve essere già stato definito. Si potrebbe per esempio numerare con una sola sequenza numerica sia i teoremi sia i lemmi sia i corollari, mentre, probabilmente, si preferirebbe numerare separatamente le definizioni, le congetture, eccetera.

Se non si specificano i contatori facoltativi, questi comandi definiscono un nuovo contatore con lo stesso nome dello pseudo ambiente che può venire stampato usando il comando `\the`*nome*. Nella stessa maniera i comandi `\label` e `\ref` usano il nuovo nome del contatore per rendere accessibile ai riferimenti incrociati anche questi enunciati. Se invece si specifica il contatore ⟨*numerato come*⟩ non viene introdotto nessun nuovo contatore ma si usa quello già esistente. Infine se si specifica il nome del ⟨*contatore dominante*⟩, il comando `\the`*nome* produrrà

in stampa il numero del contatore relativo allo pseudo ambiente preceduto dal numero stampato del ⟨*contatore dominante*⟩; se per esempio il ⟨*contatore dominante*⟩ fosse il paragrafo, allora il tredicesimo enunciato de quarto paragrafo del decimo capitolo verrà stampato nella forma 10.4.13.

E.7.4 Gestione dei numeri

A parte il modo di scrivere i numeri già visto nella pagina E.4.3, LaTeX consente di fare semplici conti con numeri interi e/o con le misure delle lunghezze.

`\newcounter` con la sintassi

> `\newcounter{`⟨*nome*⟩`}[`⟨*contatore dominante*⟩`]`
>
> permette di definire un nuovo contatore per numeri interi chiamato ⟨*nome*⟩, asservito, e quindi azzerato, quando il ⟨*contatore dominante*⟩ viene incrementato. Il ⟨*nome*⟩ e il ⟨*contatore dominante*⟩ hanno lo stesso significato descritto per i teoremi.

`\setcounter` con la sintassi

> `\setcounter{`⟨*nome*⟩`}{`⟨*valore*⟩`}`
>
> inserisce la quantità numerica ⟨*valore*⟩ nel contatore che ha quel ⟨*nome*⟩.

`\addtocounter` con la sintassi

> `\addtocounter{`⟨*nome*⟩`}{`⟨*valore*⟩`}`
>
> aggiunge il ⟨*valore*⟩ specificato al contenuto del contatore ⟨*nome*⟩. Il ⟨*valore*⟩ può anche essere negativo, quindi di fatto LaTeX esegue una somma algebrica.

`\value` con la sintassi

> `\value{`⟨*nome*⟩`}`
>
> recupera il contenuto del contatore ⟨*nome*⟩ per passarlo ad altri comandi o per rendere stampabile il numero contenuto dentro a quel contatore.

`\stepcounter` con la sintassi

> `\stepcounter{`⟨*nome*⟩`}`
>
> incrementa il contatore ⟨*nome*⟩ di una unità.

`\refstepcounter` con la sintassi

> `\refstepcounter{`⟨*nome*⟩`}`
>
> oltre ad incrementare il contatore di una unità lo rende accessibile al comando `\label` così che si possa fare riferimento ai valori del contatore mediante le chiavi usate da `\label` e `\ref`.

E.7.5 Il pacchetto **ifthen**

Il pacchetto **ifthen** permette di accedere ai comandi primitivi di controllo del flusso delle informazioni di cui TEX è capace. I comandi messi a disposizione da questo pacchetto sono i seguenti.

`\ifthenelse` con la sintassi

> `\ifthenelse{`⟨*test*⟩`}{`⟨*esegui quando è vero*⟩`}{`⟨*esegui quando è falso*⟩`}`

esegue il ⟨*test*⟩ e se questo ⟨*test*⟩ restituisce il valore 'vero' allora vengono passati al flusso di informazioni da elaborare i token che formano il contenuto del primo argomento dopo il test, ⟨*esegui quando è vero*⟩. Altrimenti vengono eseguiti i token che formano il testo di ⟨*esegui quando è falso*⟩.

I ⟨*test*⟩ che si possono eseguire sono i seguenti.

`\equal` con la sintassi

> `\equal{`⟨*stringa*$_1$⟩`}{`⟨*stringa*$_2$⟩`}`

confronta due stringhe e se sono assolutamente identiche il test è vero, altrimenti è falso.

`\lengthtest` con la sintassi

> `\lengthtest{`⟨*lungh*$_1$ `operatore` *lungh*$_2$⟩`}`

confronta due lunghezze (generalmente almeno una delle due è contenuta in un registro-lunghezza) e restituisce il valore 'vero' se le due lunghezza stanno nella relazione implicata dall'`operatore`. questo può essere un solo segno matematico fra `=`, `>`, oppure `<`.

`\isodd` con la sintassi

> `\isodd{`⟨*numero*⟩`}`

controlla se un numero (generalmente contenuto in un contatore) sia dispari.

`\boolean` controlla lo stato di una variabile booleana; la sintassi per gestire queste variabili è la seguente.

> `\newboolean{`⟨*variabile booleana*⟩`}`
> `\setboolean{`⟨*variabile booleana*⟩`}{`⟨*stato*⟩`}`
> `\boolean{`⟨*variabile booleana*⟩`}`

dove ⟨*stato*⟩ è una delle due parole `true` (vero) oppure `false` (falso).

In realtà `\boolean` è in grado di verificare lo stato anche delle variabili booleane interne a LATEX, e anche di quelle definite con i comandi elementari di TEX o di plain TEX. Con quest'ultimo si definisce un nuovo comando logico con `\newif` il quale accetta come argomento il comando per disteso e contemporaneamente definisce due altri comandi per impostare le corrispondenti variabili booleane. In pratica, se si volesse definire una nuova variabile booleana '`test`', con il pacchetto **ifthenelse** si dovrebbero usare i comandi

```
\newboolean{test}
...
\setboolean{test}{true}
...
\ifthenelse{\boolean{test}}{\textbf{Pippo}}{\textit{Pluto}}
```

Se invece si usassero i comandi di plain TeX (accessibili anche quando si usa LaTeX) si dovrebbe scrivere

```
\newif{\iftest}
...
\testtrue
...
\iftest\textbf{Pippo}\else\textit{Pluto}\fi
```

Si è fatto questo esempio non tanto per invitare ad usare i comandi elementari di plain TeX, quanto per permettere di capire come funzionano i test elementari che si trovano scritti a piene mani nei comandi definiti nei file di formato, di classe e di estensione.

`\and` `\or` e `\not` permettono di mettere insieme diverse frasi logiche da collegare fra di loro mediante questi connettori; l'intera frase comprendente i connettori deve essere racchiusa fra `\(` e `\)`.

`\whiledo` consente di descrivere e realizzare un ciclo 'while'; la sintassi è

`\whiledo{`⟨*test*⟩`}{`⟨*ciclo*⟩`}`

Questo comando ripete il ⟨*ciclo*⟩ fino a quando il ⟨*test*⟩ diventa falso. Va da sé che, prima di iniziare la ripetizione di ⟨*ciclo*⟩, gli elementi da cui dipende ⟨*test*⟩ devono essere inizializzati in modo tale che ⟨*test*⟩ sia vero. Il ⟨*ciclo*⟩ deve contenere delle istruzioni o dei comandi che prima o poi rendano il ⟨*test*⟩ falso, altrimenti LaTeX entra in un ciclo infinito e non ne esce più.

E.8 Figure, tabelle ed altri oggetti flottanti

I comandi per rendere flottanti le figure e le tabelle sono:

E.8.1 Figure e tabelle

`\begin{figure}[`⟨*posizione*⟩`]` ⟨*figura*⟩ `\end{figure}`
`\begin{figure*}[`⟨*posizione*⟩`]` ⟨*figura*⟩ `\end{figure*}`
`\begin{table}[`⟨*posizione*⟩`]` ⟨*tabella*⟩ `\end{table}`
`\begin{table*}[`⟨*posizione*⟩`]` ⟨*tabella*⟩ `\end{table*}`

I comandi senza asterisco vanno bene sia componendo ad una colonna, sia componendo a due colonne. I comandi con l'asterisco inseriscono l'oggetto a larghezza piena in testa ad una pagina composta a due colonne; per altro componendo ad una colonna si può usare indifferentemente la versione con o quella senza asterisco.

Le posizioni possibili sono

`t`	in testa alla pagina
`b`	al piede della pagina
`h`	'qui' se possibile...
`p`	in una pagina di soli oggetti flottanti

Con tutte le classi standard, se l'argomento facoltativo ⟨*posizione*⟩ non è specificato, le posizioni di default sono `tbp`.

La lista delle posizioni in realtà accetta anche `!` che invita LaTeX a 'mettercela tutta' per collocare l'oggetto nel posto più vicino a dove è stato definito. LaTeX, durante la costruzione della pagina di uscita, segue dei criteri di ottimalità per collocare gli oggetti flottanti che eventualmente sono accodati per essere emessi appena possibile. Questi criteri implicano il rispetto di certe frazioni minime o massime di spazio dedicato a questa o quella parte del documento, nonché al numero massimo di oggetti che possono essere collocati in una pagina. Le regole che LaTeX segue sono descritte qui di seguito, ma è chiaro che se si vuole che LaTeX faccia diversamente da come è stato programmato, bisogna comprendere appieno queste regole e bisogna capire come funzionano i parametri massimi o minimi che vi presiedono.

1. LaTeX colloca un oggetto flottante nel primo posto che non viola le regole seguenti, salvo che se è specificato il codice `h`, questo ha la precedenza sul codice `t`.

2. LaTeX non collocherà mai un oggetto flottante in una pagina che viene prima di quella dove compare il testo del file sorgente che attornia l'ambiente di flottaggio.

3. LaTeX colloca le figure in ordine e lo stesso fa con le tabelle, per cui non può succedere che la figura 22 appaia prima della figura 21.

4. LaTeX colloca gli oggetti flottanti solamente nelle posizioni specificate nell'argomento facoltativo ⟨*posizione*⟩ o, in mancanza di questo, in una delle tre posizioni di default `tbp`.

5. LaTeX non collocherà mai un oggetto flottante in una pagina che non contiene abbastanza spazio, quindi non produrrà mai una '`Overfull vbox`' a causa degli oggetti flottanti.

6. I vincoli imposti dai parametri stilistici tipici di ogni classe non vengono mai violati; quando viene specificato `!` vengono rilassati i vincoli imposti dai parametri che contengono sia testo sia oggetti flottanti, mentre LaTeX continua a rispettare i vincoli imposti alle pagine che contengono solamente oggetti flottanti. Per queste pagine di soli oggetti flottanti le regole vengono ignorate quando si emettono i comandi `\clearpage` oppure `\cleardoublepage`, e, ovviamente, quando si incontra `\end{document}`, perché questi comandi e la specifica che il documento è terminato ordinano a LaTeX di svuotare le code di tutto ciò che si trova ancora in memoria.

Quando si specifica la ⟨*posizione*⟩ bisogna dare abbastanza possibilità a LaTeX di fare il suo mestiere, altrimenti l'oggetto che non ha altre possibilità che una sola, blocca le code fino alla fine del documento o alla prima esecuzione di `\clearpage` o `\cleardoublepage`; questi comandi vengono emessi automaticamente quando si inizia un nuovo capitolo, ma, a parte che è brutto vedere tutte

o quasi le figure di un capitolo accumulate alla sua fine, bisogna ricordare che la memoria di LaTeX è programmata per memorizzare solamente 18 oggetti flottanti; se dovessero accodarsene di più, quelli in eccesso verrebbero persi e verrebbe emesso un messaggio che avvisa del fatto, ma è una magra consolazione.

I comandi che si possono usare dentro agli ambienti di flottaggio tipicamente sono

`\caption` con la sintassi

> `\caption[`⟨*didascalia breve*⟩`]{`⟨*didascalia*⟩`}`
>
> dove se non si specifica la ⟨*didascalia breve*⟩ questa viene automaticamente resa identica alla didascalia 'lunga'; potrebbe non essere una buona idea quella di inviare alla lista delle figure o delle tabelle l'intera didascalia, specialmente se questa contiene uno o più periodi dopo il primo che svolge il compito di titolo, mentre i periodi successivi svolgono il compito di fornire maggiori delucidazioni sull'oggetto specifico.

`\label` può essere usato, anzi è conveniente che sia usato per ogni figura; il comando deve essere collocato dopo il comando `\caption`, perché finché questo non viene eseguito, all'oggetto flottante non è ancora stato assegnato un numero.

`\suppressfloats` con la sintassi

> `\suppressfloats[`⟨*posizione*⟩`]`
>
> può essere collocato fuori degli ambienti di flottaggio al fine di evitare che LaTeX metta altri oggetti nella stessa ⟨*posizione*⟩ specificata come argomento facoltativo; se non si specifica nulla, tutti gli oggetti flottanti sono esclusi dalla pagina corrente; Se però nella ⟨*posizione*⟩ del comando di apertura degli ambienti *figure* o *table* compare `!`, allora `\suppressfloats` viene ignorato.

I parametri dimensionali e numerici che regolano il deflusso degli oggetti flottanti dalle rispettive code sono i seguenti.

`topnumber` è il nome del contatore che contiene il numero *massimo* di oggetti flottanti che possono essere collocati in testa alla pagina.

`\topfraction` è la *massima* frazione di pagina destinata agli oggetti flottanti in testa ad una pagina, se questa contiene anche del testo.

`bottomnumber` è il contatore che contiene il numero *massimo* di oggetti flottanti in calce alla pagina.

`\bottomfraction` è la *massima* frazione di pagina destinata agli oggetti flottanti in calce ad una pagina, se questa contiene anche del testo.

`\totalnumber` è il contatore che contiene il *massimo* numero di oggetti flottanti che possono comparire in una pagina di testo, indipendentemente dal fatto che che siano in testa o in calce.

`\textfraction` per una pagina che contenga anche del testo questa è la frazione *minima* della pagina destinata al testo.

`\floatpagefraction` in una pagina di soli oggetti flottanti rappresenta la *minima* frazione di pagina che deve essere occupata da questi oggetti; se cioè si specifica il parametro di posizione `p` per un oggetto flottante e questo è troppo piccolo, esso viene trattenuto in memoria finché non si trova un altro oggetto flottante dello stesso genere e con lo stesso attributo di posizione che assieme possano superare questo valore minimo specificato.

`dbltopnumber` il nome del contatore che contiene il numero *massimo* di oggetti flottanti a piena pagina da mettere in testa ad una pagina con il testo composto su due colonne.

`\dbltopfraction` la *massima* frazione di pagina a giustezza piena da destinare agli oggetti flottanti in testa alla pagina quando si compone a due colonne.

`\dblfloatpagefraction` è la *minima* frazione di pagina da occupare con oggetti flottanti a giustezza piena quando si compone una pagina di soli oggetti flottanti in un testo composto a due colonne. È, insomma, l'analogo di `\floatpagefraction` che, invece, vale quando si compone il testo ad una sola colonna.

`\floatsep` è la distanza *minima* da riempire con un contrografismo incolore fra due oggetti flottanti consecutivi.

`\textfloatsep` è la distanza *minima* da interporre fra gli oggetti flottanti in testa o in calce e il testo adiacente.

`\intextsep` è la distanza *minima* fra un oggetto flottante a centro pagina e il testo che lo precede e il testo che lo segue.

`\dblfloatsep` è la distanza *minima* fra due oggetti flottanti consecutivi in testa ad una pagina composta a due colonne.

`\dbltextfloatsep` è la distanza *minima* posta fra gli oggetti flottanti in testa ad una pagina composta a due colonne e il testo sottostante.

I contatori si impostano con i comandi specifici, in particolare `\setcounter`; le frazioni `\...fraction` sono valori decimali, in generale fratti e minori dell'unità che vengono conservati dentro le macro con i rispettivi nomi, e quindi tutti si reimpostano, per modificare i valori di default, mediante `\renewcommand`. I separatori `\...sep` sono lunghezze e si impostano con il comando `\setlength`. Ogni classe definisce i suoi particolari valori per questi parametri; per la classe *`book`* composta in corpo 10 (come questo testo) i parametri sono riportati nella tabella E.3.

Va detto che i 'Numeri' e le 'Frazioni' indicati nelle tabelle E.3 non devono necessariamente essere consistenti; sembra strano, ma è comprensibile. Per i 'Numeri', per esempio, `totalnumber` sembra essere la somma di `topnumber` e `bottomnumber`; se però `totalnumber` fosse posto al valore 2, questo vorrebbe dire che *al massimo* ci possono essere 2 oggetti flottanti in testa e uno in calce, ma in ogni caso non possono essere tutti e tre presenti, perché nella pagina ce ne possono essere *al massimo* due. Lo stesso vale per le frazioni; esse rappresentano

Numeri		Frazioni	
`topnumber`	2	`\topfraction`	0,7
`bottomnumber`	1	`\bottomfraction`	0,3
`totalnumber`	3	`\textfraction`	0,2
`dbltopnumber`	2	`\floatpagefraction`	0,5
		`\dbltopfraction`	0,7
		`\dblfloatpagefraction`	0,5

Separatori	
`\floatsep`	`12pt plus 2pt minus 2pt`
`\textfloatsep`	`20pt plus 2pt minus 4pt`
`\intextsep`	`12pt plus 2pt minus 2pt`
`\dblfloatsep`	`12pt plus 2pt minus 2pt`
`\dbltextfloatsep`	`20pt plus 2pt minus 4pt`

Tabella E.3: Parametri nella classe *book* composta in corpo 10 per gestire gli oggetti flottanti

dei limiti per delle disuguaglianze; questi limiti non debbono necessariamente avere per somma l'unità; l'importante è che queste disuguaglianze siano tutte rispettate nell'ordine di preferenza che LaTeX assegna alla collocazione degli oggetti flottanti.

Il compositore si ricordi, specialmente per quel che riguarda le frazioni, che nel computo dell'ingombro dell'oggetto flottante bisogna tenere conto anche dello spazio destinato alla didascalia e del suo spazio di separazione. Spesso il compositore è deluso dalla rigidezza con cui LaTeX gestisce gli oggetti flottanti; in realtà LaTeX, come tutti i programmi, si comporta come gli è stato prescritto, in particolare, in questo caso, come richiesto dai 'Numeri', dalle 'Frazioni' e dai 'Separatori' che compaiono nella tabella E.3. Se non piacciono quei valori, li si può cambiare, ma il compositore stia attento che la cura non sia peggio del male. Piuttosto egli dedichi la sua attenzione al dimensionamento degli oggetti flottanti, all'eliminazione di spazi inutili al loro contorno, all'eliminazione delle parti inutili delle fotografie che si trovano spessissimo ai margini laterali o verticali; insomma usi appropriatamente le possibilità offerte dalle opzioni di `\includegraphics` perché è con quelle che si curano davvero sia il deflusso degli oggetti flottanti dalle rispettive code, sia la qualità del documento prodotto.

E.8.2 Note marginali

I comandi che portano alla composizione e alla collocazione delle note marginali sono i seguenti:

```
\marginpar[⟨nota di sinistra del testo⟩]{⟨nota di destra del testo⟩}
\reversemarginpar
\normalmarginpar
```

La posizione normale delle note quando si compone ad una colonna è il margine esterno; quando si compone a due colonne è il margine adiacente alla colonna alla quale la nota si riferisce, quindi il margine di sinistra per la colonna di sinistra e

il margine di destra per la colonna di destra; e questo avviene indipendentemente dal fatto che ci si trovi in una pagina di sinistra o di destra. `\reversemarginpar` scambia i margini per le note composte ai margini di testi composti ad una sola colonna e `\normalmarginpar` ripristina la collocazione di default. Questi due comandi sono inattivi quando si compone a due colonne.

Per il testo delle note che dovrebbero apparire a sinistra del testo a cui si riferiscono potrebbe essere applicata la dichiarazione `\raggedleft` per avere una composizione in bandiera giustificata a destra. Tuttavia questo è consigliabile solo quando le annotazioni marginali sono sempre e consistentemente molto brevi in modo da non superare una riga. Cambiare giustificazione da nota a nota sarebbe una caduta di uniformità e di stile. D'altra parte le note di più righe giustificate solo a destra sono leggermente più laboriose da leggere rispetto alle note giustificate solo a sinistra; il compositore ci pensi bene prima di decidere come comporre le note; eventualmente si crei una macro che componga sempre entrambi i testi di sinistra e di destra uniformemente con lo stesso stile, in modo da non dover operare a mano così da evitare i possibili errori di disuniformità.

I parametri che governano la collocazione delle note, a parte il margine 'normale' o 'scambiato', sono:

`\marginparwidth` è la giustezza delle note marginali e deve essere evidentemente minore della larghezza del margine fisico al lato del testo.

`\marginparsep` è lo spazio di separazione fra il testo e il blocchetto della nota marginale.

`\marginparpush` è lo spazio con cui una nota marginale viene spostata in basso rispetto alla fine della nota marginale precedente; talvolta questo obbliga a portare la nota nella pagina successiva, ma viene emesso un avvertimento sullo schermo e nel file `.log`. Per riparare queste situazioni non tanto desiderabili, bisogna necessariamente riformulare il testo principale a cui la nota in questione si riferisce, oppure, bisogna scorciare il testo della nota precedente.

E.9 Incolonnamenti

E.9.1 L'ambiente **tabbing**

Nel testo non se ne è nemmeno parlato, perché si ritiene che questo ambiente sia una reminiscenza degli incolonnamenti eseguiti in dattilografia; quando TEX 78 nacque, le macchine da scrivere meccaniche ed elettriche erano ancora diffusissime e le tabulazioni eseguite con il tasto di tabulazione, nonché con gli arresti di tabulazione erano familiari a tutti. Tuttavia l'ambiente *tabbing*, nato, forse, proprio per affidarsi a qualcosa di noto a qualunque dattilografo, presenta dei notevoli inconvenienti, primo fra i quali quello di essere un ambiente fragile e che non può essere annidato dentro altri ambienti.

La sintassi è quella comune a qualunque ambiente:

`\begin{tabbing}` ⟨*testo da tabulare*⟩ `\end{tabbing}`

I comandi per gestire gli arresti di tabulazione sono i seguenti.

`\=` serve per impostare un arresto di tabulazione; spesso conviene scrivere una riga modello da *non* comporre se si ricorre al comando seguente.

`\kill` non compone la riga che precede questa parola chiave, ma conserva le informazioni di impostazione degli arresti di tabulazione che la riga conteneva.

`\>` serve per spostare la composizione al successivo arresto di tabulazione.

`\\` comincia una nuova riga riportando il contatore degli arresti di tabulazione al valore iniziale, generalmente zero.

`\+` incrementa di una unità il valore iniziale del contatore di tabulazione così che da questo momento in poi e fino ad ordine contrario, tutte le righe risultano indentate e incolonnate sotto un arresto presente nella riga precedente.

`\-` decrementa di una unità il valore iniziale del contatore di tabulazione; contrasta l'effetto di `\+`.

`\<` torna indietro di un arresto di tabulazione e può essere usato solamente all'inizio di una riga.

`\'` serve per spostare tutto il contenuto di una 'colonna' a filo del margine destro del suo campo di tabulazione.

`\`` serve per comporre il contenuto della sua colonna a filo del margine sinistro del suo arresto di tabulazione.

`\pushtabs` manda in memoria le posizioni degli attuali arresti di tabulazione, consentendo di impostarne di diversi.

`\poptabs` serve per richiamare dalla memoria un insieme di arresti di tabulazione precedentemente memorizzati.

`\a= \a'` e `\a`` agiscono come `\=`, `\'` e `\`` per comporre gli accenti macron (lungo), acuto e grave rispettivamente, perché questi comandi, come si vede sopra, sono stati ridefiniti all'interno dell'ambiente per svolgere funzioni diverse. Il compositore che usa il pacchetto **inputenc** non se ne preoccupi e continui ad usare tranquillamente le lettere accentate della sua tastiera, perché quel pacchetto provvede alla compatibilità senza che il compositore debba preoccuparsene più di tanto. Il compositore deve purtroppo occuparsene se e solo se la sua tastiera è priva di segni accentati e se il suo shell editor non consente di usare combinazioni di tasti per inserire direttamente nel file `.tex` i segni accentati, oppure, infine, se ha impostato il suo shell editor in modo che salvi i file che usino solo caratteri ASCII a 7 bit, così da assicurare la massima portabilità 'cross-platform'.

E.9.2 Gli ambienti array e tabular

Le sintassi sono:

```
\begin{array}[⟨posizione⟩]{⟨colonne⟩} ⟨righe⟩ \end{array}
\begin{tabular}[⟨posizione⟩]{⟨colonne⟩} ⟨righe⟩ \end{tabular}
\begin{tabular*}{⟨larghezza⟩}[⟨posizione⟩]{⟨colonne⟩} ⟨righe⟩ \end{tabular*}
```

La ⟨*posizione*⟩ serve per stabilire l'allineamento della tabella o della matrice con il 'testo' circostante; la ⟨*larghezza*⟩ serve per imporre una larghezza specificata alla tabella. I descrittori delle ⟨*colonne*⟩ sono certi codici che ora si rivedranno; le ⟨*righe*⟩ sono i contenuti delle celle orizzontali contenenti il testo o la matematica da comporre. *array* si usa solo in ambiente matematico; gli ambienti *tabular* e *tabular** solo in modo testo; sia gli uni sia gli altri ambienti possono contenere singole celle contenenti 'testo' dell'altra specie, ma bisogna specificarlo per ciascuna di queste celle; in ambiente *tabular* basta usare gli appositi comandi per passare alla matematica; in ambiente *array* si possono usare i segni di dollaro che agiscono da interruttori/deviatori; se si è già in modo matematico (come avviene per ogni cella di un *array*) il primo segno di dollaro passa in modo testo e il secondo riporta al modo matematico.

I codici di posizione sono

`t` serve per allineare la riga di testa con il testo circostante.

`b` serve per allineare la riga di base con quella del testo circostante.

`c` non necessita di essere espresso, perché l'allineamento centrato è quello di default.

I descrittori delle ⟨*colonne*⟩ e i loro separatori sono i seguenti.

`l` colonna con i contenuti allineati a sinistra.

`c` colonna con i contenuti centrati.

`r` colona con i contenuti allineati a destra.

`|` filetto verticale di separazione fra colonne adiacenti.

`\vline` lo stesso filetto quando deve essere inserito in una *@-espressione*.

`@-{`⟨*testo*⟩`}` *@-espressione*; si tratta di un particolare costrutto con il quale si specifica che ⟨*testo*⟩ sostituisce completamente il separatore ordinario fra due colonne adiacenti. Questo significa che se i contenuti delle due celle adiacenti devono essere in qualche modo distanziati, allora gli spaziatori devono essere inseriti dentro la *@-espressione*. Un elemento importante delle *@-espressioni* sono le dichiarazioni di spaziatura 'infinita' da usare con l'ambiente *tabular** per consentire che queste particolari tabelle si possano estendere fino a raggiungere la larghezza specificata.

`\extracosep{`⟨*larghezza*⟩`}` specificato dentro una *@-espressione* serve a dichiarare che lo spaziatore con la ⟨*larghezza*⟩ esplicitata venga inserito a sinistra del contenuto di tutte le celle *che seguiranno* sulla sessa riga; quindi *non* di questa e di tutte le celle che seguiranno, ma solo di quelle che seguiranno. Tipicamente questo comando viene inserito nella prima *@-espressione* che compare a sinistra della prima cella di una riga; quindi questa cella non verrà mai allargata per raggiungere la giustezza desiderata.

`\fill` è una larghezza naturalmente nulla ma dotata di allungamento infinito, tipicamente usata come argomento di `\extracolsep`.

`p{`⟨*larghezza*⟩`}` serve per descrivere una colonna composta come un (breve) capoverso la cui prima riga è allineata con quella delle celle adiacenti e avente una giustezza pari a ⟨*larghezza*⟩. Il testo delle colonne composte con questo descrittore non può contenere direttamente il comando `\\` perché LaTeX non potrebbe sapere se esso si riferisce alla cella o all'intera riga di celle. Perciò è necessario racchiudere questo comando dentro un ambiente, come *minipage* o un altro *tabular*, oppure dentro una scatola `\parbox` esplicita; oppure dentro lo scopo di comandi come `\centering`, `\raggedright` oppure `\raggedleft` purché a loro volta siano contenuti dentro un gruppo delimitato da parentesi graffe oppure siano contenuti all'interno di altri ambienti.

`*{`⟨*numero*⟩`}{`⟨*descrittori*⟩`}` è una forma abbreviata per ripetere ⟨*numero*⟩ volte le stessa sequenza di ⟨*descrittori*⟩ delle colonne. Una **-espressione* ne può contenere un'altra; si consiglia di non eccedere con queste forme stenografiche innestate l'una nell'altra altrimenti dopo un po' non si capisce più che cosa si sia specificato. Piuttosto è meglio rifarsi alle possibilità offerte dal pacchetto **array** e dal comando `\newcolumntype`.

I comandi che si possono usare dentro gli ambienti di incolonnamento sono i seguenti.

`\multicolumn` con la sintassi

`\multicolumn{`⟨*numero*⟩`}{`⟨*descrittore*⟩`}{`⟨*cella*⟩`}`

serve per comporre un'unica ⟨*cella*⟩ con il ⟨*descrittore*⟩ specificato che occupi ⟨*numero*⟩ celle adiacenti. Il ⟨*descrittore*⟩ oltre alla collocazione con una delle varie lettere chiave (compresa la `p`) può contenere filetti verticali e persino *@-espressioni*. Normalmente il contenuto di `\multicolumn` rimpiazza completamente tutte le ⟨*numero*⟩ celle adiacenti, compresi i loro separatori destri; quindi il separatore sinistro non dovrebbe mai essere indicato come contenuto del ⟨*descrittore*⟩ a meno che il gruppo di celle non comprenda anche la prima cella di una riga.

`\hline` serve per tirare un filetto orizzontale attraverso tutta la tabella o tutta la matrice. Esso può essere eseguito solo come primo elemento di una tabella, oppure dopo il comando di fine riga `\\`, oppure dopo un altro comando `\hline`; in quest'ultimo caso vengono tracciati due filetti ravvicinati attraverso tutta la tabella.

`\cline{`⟨*col*$_1$*-col*$_2$⟩`}` serve per tracciare un filetto orizzontale solo sotto alle colonne ⟨*col*$_1$*-col*$_2$⟩; per esempio in una tabella a 7 colonne, il comando `\cline{2-6}` traccia un filetto orizzontale sotto le 5 colonne centrali; non si possono inserire due identici comandi `\cline` uno di seguito all'altro per raddoppiare il filetto, ma si possono usare per 'sottolineare' colonne distinte, per esempio nella tabella a 7 colonne dell'esempio precedente `\cline{2-3} \cline{5-6}` 'sottolinea' solo le colonne 2 e 3 con un filetto e, allineato con il primo, un secondo filetto 'sottolinea' le colonne 5 e 6.

I parametri compositivi di tabelle e matrici sono i seguenti.

`\arraycolsep` è metà dello spazio di separazione fra due celle consecutive di una matrice.

`\tabcolsep` e metà dello spazio di separazione fra due celle consecutive di una tabella.

`\arrayrulewidth` è lo spessore dei filetti orizzontali o verticali che si possono inserire in tabelle e matrici.

`\doublerulesep` è la spaziatura verticale o orizzontale fra due filetti consecutivi orizzontali o verticali.

`\arraystretch` è una macro, che può venire ridefinita con `\renewcommand`, e che rappresenta il fattore di scala con cui viene ingrandito o rimpicciolito lo strut che si trova sempre nella prima cella di una tabella o matrice in ogni riga. Il suo valore di default è l'unità.

E.10 I file ausiliari e i loro comandi

E.10.1 I file del sistema TEX

LATEX, come per altro anche plain TEX, fa riferimento ad un gran numero di file dei quali spesso il compositore non si accorge nemmeno. A parte i file sorgente con estensione `.tex` o `.ltx`, ci sono anche i file con le estensioni seguenti; il nome proprio dei file corrispondenti ad un medesimo documento è costantemente il nome proprio del file principale e questo nome è conservato dentro la variabile `\jobname` che può essere usata anche dal compositore, o meglio, dal programmatore che scrive le macro.

`.aux` serve per conservare tutte le informazioni che riguardano i riferimenti incrociati; quando si usano i comandi `\include`, questi file conservano anche tutte le informazioni relative al file incluso in modo da poterne simulare la compilazione quando questo file non sia incluso nella lista di file da compilare contenuta nell'argomento di `\includeonly`. Il comando `\nofiles` sopprime la scrittura di tutti i file di questo tipo.

`.bbl` questo file viene scritto da BIBTEX, ma viene poi letto da LATEX per comporre la bibliografia quando si usi il comando `\bibliography`.

`.dvi` è il formato di default dell'uscita compilata di tutti i programmi elaborati con il sistema TEX; esso fa riferimento ai font a matrici di pixel e generalmente il programma per rendere leggibile agli umani questo file è incluso nella distribuzione del sistema TEX; pdfLATEX, che viene usato per produrre direttamente l'uscita in formato PDF, è in grado di produrre l'uscita in formato DVI se si pone al suo inizio `\pdfoutput=0`.

`.glo` è il file prodotto da LATEX quando sia attivo il comando `\makeglossary`; esso contiene tutte le voci `\glossaryentry` prodotte dai vari comandi `\glossary` inseriti nei file sorgente.

`.idx` è il file prodotto da LATEX quando sia attivo il comando `\makeindex`; esso contiene tutte le voci `\indexentry` prodotte dai vari comandi `\index` inseriti nel file sorgente.

`.ind` questo file viene scritto dal programma makeindex ma viene riletto da LaTeX quando deve comporre l'indice analitico.

`.lof` contiene le informazioni per comporre l'indice delle figure.

`.log` contiene tutta le registrazione di quanto è successo durante l'esecuzione di LaTeX, in particolare le informazioni relativamente dettagliate concernenti gli errori e gli avvertimenti, i font usati, quelli che sono stati creati al volo, quelli mancanti, quelli sostituiti, eccetera.

`.lot` contiene le informazioni per comporre la lista delle tabelle.

`.toc` contiene le informazioni per comporre l'indice generale.

E.10.2 I riferimenti incrociati

I comandi per assegnare una parola chiave agli oggetti da citare e/o per richiamarne il valore o la pagina sono:

```
\label{⟨chiave⟩}
\ref{⟨chiave⟩}
\pageref{⟨chiave⟩}
```

Il comando `\label` è fragile; sarebbe meglio, anche se è possibile, non inserirlo mai all'interno di altri comandi, nemmeno i comandi di sezionamento o i comandi per le didascalie. I contatori che possono essere citati con la chiave sono quelli che sono stati incrementati con `\refstepcouinter`; in ogni caso `\label` associa alla chiave il valore dell'ultimo contatore in ordine di tempo che è stato incrementato con quel comando.

E.10.3 Bibliografia e citazioni

Salvo disporre di file bibliografici esterni e di fare uso del programma BibTeX, LaTeX dispone dell'ambiente *thebibliography* per comporre gli elenchi bibliografici e assegnare loro una chiave di riferimento.

```
\begin{thebibliography}{⟨cifre⟩}
\bibitem[⟨etichetta⟩]{⟨chiave⟩} ⟨voce bibliografica⟩
...
\end{thebibliography}
```

Le voci bibliografiche vengono richiamate con il comando

```
\cite[⟨testo⟩]{⟨chiave⟩}
oppure
\cite{⟨lista di chiavi⟩}
```

La ⟨*chiave*⟩ è una informazione interna di LaTeX che serve a LaTeX stesso e al compositore per riferirsi ad una particolare voce bibliografica. Quando questa viene citata, viene citata o con l'⟨*etichetta*⟩, se è stata specificata, oppure con il numero progressivo dell'elenco bibliografico; in ogni caso numero o etichetta vengono racchiusi entro parentesi quadre.

Con `\cite` si possono fare citazioni multiple, semplicemente specificando una ⟨*lista di chiavi*⟩ formata dalle chiavi specifiche dei riferimenti da citare separate da virgole; per questo motivo le chiavi possono essere formate con qualunque carattere, esclusa la virgola. Se si cita un'opera sola, la citazione contiene l'⟨*etichetta*⟩, o il numero, seguiti da ⟨*testo*⟩; per esempio: avendo attribuito ad un riferimento l'etichetta `TUG2007`, il comando di citazione `\cite[cap.~2]{TUG2007}` produrrà qualcosa come [12, cap. 2].

Per la bibliografia, specialmente se contiene numerose voci, è consigliabile servirsi del programma BibTeX, che garantisce la composizione uniforme delle varie citazioni e, tramite file di estensione, consente anche di usare stili di citazione diversi, come per esempio lo stile 'autore-anno'.

E.10.4 Suddivisione del file sorgente

Non è conveniente scrivere un unico file sorgente, specialmente se il documento da comporre contiene molte pagine, molti capitoli, eccetera. Conviene predisporre un 'masterfile' che contenga solo il preambolo e la lista dei file componenti.

```
\documentclass[⟨opzioni⟩]{⟨classe⟩}
\usepackage[⟨opzioni⟩]{⟨pacchetto⟩}[⟨data⟩]
...
\includeonly{⟨lista di file⟩}
\begin{document}
\include{⟨primo file⟩}
\include{⟨secondo file⟩}
...
\end{document}
```

La ⟨*lista di file*⟩ è un elenco di nomi di file separati da virgole, ma senza spazi spuri prima o dopo le virgole. La sequenza di comandi `\include`, ognuno con il suo nome di file, esegue la compilazione dei file, tranne di quelli che *non* sono elencati nella ⟨*lista di file*⟩. Al limite si può compilare un file alla volta; i file già compilati, anche se non vengono più elaborati, hanno lasciato le loro tracce nei rispettivi file `.aux`, quindi, a meno che non siano intervenute modifiche, LaTeX riesce a risolvere tutti i riferimenti incrociati e a mantenere la coerenza dei numeri delle pagine e di tutti i contatori specificati con i comandi specifici di LaTeX.

È chiaro che prima di terminare la lavorazione di un documento è importante eseguire la compilazione di tutto il documento, cosicché si è completamente sicuri che tutti i riferimenti e tutti i valori dei contatori e delle pagine siano corretti; questo lavoro, comunque va fatto per predisporre alla fine della lavorazione la compilazione dell'indice analitico.

Ci si ricordi che ogni comando `\include` per prima cosa esegue un comando `\clearpage` che svuota le code di oggetti flottanti e comincia la compilazione del nuovo file su una pagina nuova. Se *non* è questo quello che si vuole ottenere, si faccia uso del comando

```
\input{⟨file⟩}
```

che non consente la compilazione selettiva, ma consente di proseguire la compilazione del testo dal punto preciso nel quale si trovava LaTeX nel momento in cui ha cominciato ad eseguire il comando `\input`.

Per risolvere le dipendenze di un dato documento da certi file di macro, è disponibile l'unico ambiente che può precedere il comando `\documentclass`

```
\begin{filecontents}{⟨nome file⟩}
⟨corpo del file⟩
\end{filecontents}
```

È disponibile anche l'ambiente asteriscato; la differenza consiste in questo: quando *filecontents* viene eseguito, controlla che un file con il ⟨*nome file*⟩ specificato esista; se esiste, l'ambiente non fa nulla salvo scrivere sullo schermo e nel file `.log` che il file esiste già; se invece il file non esiste, questo ambiente lo crea scrivendovi dentro il ⟨*corpo del file*⟩ e salvandolo con il nome ⟨*nome file*⟩. Nel fare questo, al fine di identificare il tipo di file e la sua origine, esso scrive anche dei commenti in stile LaTeX, cioè con un segno `%` all'inizio della riga. Se non si vogliono questi commenti si usi l'ambiente asteriscato.

Il comando `\listfiles` serve per elencare a schermo e nel file `.log` tutti i nomi dei file che vengono via via aperti (tranne quelli di servizio) in modo che si possa seguire e controllare che l'esecuzione della compilazione proceda regolarmente.

E.11 Indice analitico e glossario

Come descritto nel capitolo 15 per comporre uno o più indici analitici è opportuno servirsi del programma makeindex; nello stesso capitolo viene anche descritto come servirsi di quel programma per compilare un glossario. Tuttavia qui si richiamano i comandi necessari.

E.11.1 Indice analitico

La composizione dell'indice analitico avviene mediante l'ambiente *theindex*

```
\begin{theindex}
\input{⟨indexfile⟩.ind}
\end{theindex}
```

Più comodamente si usa il pacchetto **makeidx** e si ordina la composizione dell'indice analitico mediante il comando `\printindex`.

Però per raccogliere le voci da indicizzare bisogna usare i seguenti comandi

`\makeindex` inserito nel preambolo ordina di eseguire effettivamente la raccolta delle voci attraverso il comando `\index`, che altrimenti sarebbe completamente inerte.

`\index` serve per raccogliere le voci da indicizzare con la sintassi

```
\index{⟨voce⟩}
```

Il comando `\index` deve essere scritto senza spazi interposti alla fine della parola di testo che si vuole indicizzare. La ⟨*voce*⟩ non contiene solo la parola da indicizzare, ma anche le ulteriori informazioni di cui necessita il programma makeindex per formattare convenientemente la voce come voce principale, oppure secondaria, oppure terziaria, per scegliere il font con cui scrivere la voce, per scegliere la chiave di indicizzazione, per segnalare , nel caso di voci secondarie o terziarie, sotto quale voce primarie esse debbano essere elencate, per dare uno stile al numero di pagina, eccetera; si veda a questo proposito la documentazione del programma makeindex.

E.11.2 Glossario

Per comporre un glossario bisogna procedere in modo simile a quello che si usa per l'indice analitico; i comandi che permettono di raccogliere le voci sono

`\makeglossary` inserito nel preambolo consente l'effettiva raccolta delle voci mediante l'attivazione del comando `\glossary`, che altrimenti sarebbe totalmente inattivo.

`\glossary` serve per raccogliere le voci da inserire nel glossario.

E.12 Compilazione interattiva

LaTeX consente un certo livello di interattività durante la compilazione; a parte gli errori, che chiedono al compositore di intervenire, LaTeX consente anche di emettere dei messaggi sul terminale e di leggere le risposte del compositore.

`\typeout` con la sintassi

`\typeout{`⟨*messaggio*⟩`}`

consente di scrivere un messaggio sullo schermo del terminale; questo messaggio può avere gli scopi più diversi, ma non consente nessuna interattività; piuttosto può preparare un dialogo interattivo.

`\typein` con la sintassi

`\typein[`⟨*comando*⟩`]{`⟨*messaggio*⟩`}`

emette il ⟨*messaggio*⟩ sullo schermo; se è specificato anche il ⟨*comando*⟩ il programma si arresta e attende che l'operatore scriva qualche cosa sulla tastiera (presumibilmente quanto il ⟨*messaggio*⟩ suggerisce di scrivere fra una rosa di possibili scelte); come il compositore preme il tasto di fine-riga, il testo introdotto viene assegnato al ⟨*comando*⟩, esattamente come se il tutto eseguisse le seguenti operazioni

`\typeout{`⟨*messaggio*⟩`}`
`\newcommand{`⟨*comando*⟩`}{`⟨*testo introdotto*⟩`}`

Il ⟨*comando*⟩ deve essere una valida sequenza di controllo, cioè una stringa esclusivamente letterale preceduta dal backslash, oppure un solo segno non letterale preceduto dal backslash, oppure un carattere attivo.

E.13 Interruzione di riga e di pagina

E.13.1 Interruzione di riga

I comandi

```
\linebreak[⟨numero⟩]
\nolinebreak[⟨numero⟩]
```

consentono o vietano, rispettivamente, di interrompere una riga. Se si specifica il ⟨*numero*⟩ la loro azione è rinforzata o indebolita a seconda del valore del numero stesso. Il numero 0 rappresenta un debole incoraggiamento ad operare, mentre il numero 4 rappresenta l'indicazione di eseguire incondizionatamente l'azione specificata; l'assenza del numero equivale a specificare il numero 4. Attenzione: `\linebreak` interrompe la linea ma lascia che LaTeX cerchi di giustificarla, per cui se la gomma interparola deve essere allargata troppo, viene emesso un messaggio di `Underfull hbox`.

Una riga può venire interrotta anche con i comandi

```
\\[⟨spazio⟩]
\\*[⟨spazio⟩]
\newline
```

Essi permettono di interrompere una riga per andare decisamente a capo. Non viene eseguito nessun tentativo di giustificare la riga, nemmeno se manca poco al margine destro. La forma asteriscata impedisce un fine pagina subito dopo.

Per tutti questi comandi la possibilità di eseguire la cesura delle parole in fin di riga è essenziale. Per altro in certe circostanze non basta consentire la cesura, ma bisogna diminuire le tolleranze di composizione, cioè bisogna accontentarsi di una maggiore bruttezza di una o più righe. Ciò si ottiene con i comandi

```
\sloppy
\fussy
```

Il primo aumenta le tolleranze di composizione a valori enormi, che per LaTeX equivalgono ad infinito. Il secondo ripristina i valori normali.

E.13.2 Interruzione di pagina

I comandi

```
\pagebreak[⟨numero⟩]
\nopagebreak[⟨numero⟩]
```

Agiscono come `\linebreak` ma riferendosi alla pagina, non alla riga. Il ⟨*numero*⟩ ha lo stesso significato con 0 che indica in blando incoraggiamento ad eseguire, e 4 che ordina di eseguire incondizionatamente il comando.

Per l'interruzione di pagina non si può agire sulla cesura, ma si può temporaneamente allungare la pagina rispetto al normale:

```
\enlargethispage{⟨lunghezza⟩}
\enlargethispage*{⟨lunghezza⟩}
```

Il comando senza asterisco allunga semplicemente la pagina; per ovvi motivi ⟨*lunghezza*⟩ dovrebbe essere un multiplo intero di `\baselinskip`, dell'avanzamento di riga normale con il font di default. Tuttavia il comando asteriscato cerca di allungare la pagina di quanto specificato, ma cerca anche di stringere il più possibile gli spazi bianchi verticali al fine di non allungare la pagina, ovvero di allungarla il meno possibile. Specialmente componendo sul verso e sul recto è importante che le pagine siano della stessa altezza; se i margini laterali sono consistenti, una riga in più in una delle due pagine si nota appena, tuttavia sarebbe meglio evitarlo. Il comando asteriscato potrebbe essere la soluzione se la pagina nella quale esso compare ha abbastanza spazi bianchi.

I comandi

```
\newpage
\clearpage
\cleardoublepage
```

chiudono la pagina corrente; gli ultimi due scaricano anche le eventuali code di oggetti flottanti ancora non svuotate; il comando `\cleardoublepage` eventualmente emette anche una pagina bianca al fine di consentire di iniziare la pagina successiva sul recto. `\cleardoublepage` è tipicamente il primo comando che viene eseguito implicitamente quando si esegue il comando di inizio di un nuovo capitolo e anche quando si immette un file con `\include`.

Sono disponibili anche dei comandi imprestati da plain TEX, ma ancora definiti con LATEX:

`\smallbreak` esegue un salto pagina solo se prima era stato introdotto nel testo uno spazio verticale pari a `\smallskipamount`.

`\medbreak` esegue un salto pagina solo se prima era stato introdotto nel testo uno spazio verticale pari a `\medskipamount`.

`\bigbreak` esegue un salto pagina solo se prima era stato introdotto nel testo uno spazio verticale pari a `\bigskipamount`.

`\goodbreak` indica a LATEX che questo è un buon punto per spezzare la pagina; LATEX effettivamente esegue il salto se manca poco alla fine naturale della pagina, se, cioè non resta una pagina pesantemente mozza, ma è solo corta di pochissime righe.

`\filbreak` sempre se manca poco alla fine della pagina, questo comando indica a LATEX che può eseguire un salto pagina inserendo anche un po' di gomma verticale in modo che non vengano emessi messaggi di `Underfull vbox`; nello stesso tempo, se il punto non è buono per interrompere la pagina `\filbreak` non fa nulla.

Si consiglia di usare questi comandi solo alla fine della lavorazione di un documento per correggere a mano alcune piccole cose che LATEX da solo non è capace di gestire. I casi in cui il compositore sia costretto a ricorrere a questi comandi e a quelli di LATEX sono rarissimi in testi 'normali'; possono essere frequenti in testi che contengano molta matematica.

E.14 Lunghezze, spazi e scatole

E.14.1 Lunghezze

Le lunghezze esplicite, quelle, cioè, che si esprimono con una misura e una unità di misura, sono costituite da una sequenza di cifre decimali, separate dal separatore decimale che in teoria potrebbe anche essere la virgola; siccome nella sintassi LaTeX la virgola svolge anche altri ruoli, è preferibile usare sempre e soltanto il punto decimale quando si esprimono le misure delle lunghezze. Un numero che contenga una parte intera nulla potrebbe cominciare con il punto decimale, omettendo lo zero che dovrebbe precederlo; si sconsiglia questa abbreviazione del tutto irrilevante ma che facilita gli errori. Il numero può essere preceduto dal segno positivo o negativo; quello positivo è evidentemente ridondante; quello negativo invece è necessario. Se in seguito alla sostituzione di espressioni ed argomenti nelle macro la misura risultasse preceduta da diversi segni, il numero di segni negativi si comporta come ci si aspetterebbe, nel senso che ognuno cambia segno a quanto segue, quindi un numero dispari di segni negativi porta ad un risultato complessivamente negativo, mentre un numero pari porta ad un risultato positivo.

Unità di misura Le unità di misura accettabili da LaTeX sono le seguenti:

- `cm` centimetri.
- `mm` millimetri.
- `in` pollici.
- `pt` punti tipografici (1 pt=(1/72,27) in).
- `pc` pica (1 pc=12 pt).
- `bp` punto PostScript (1 bp=(1/72) in).
- `sp` *scaled point* (la frazione 2^{-16} di un punto tipografico).
- `dd` punto didot.
- `cc` cicero (1 cc=12 dd).
- `ex` *x-height*, occhio del carattere corrente, altezza della lettera 'x'.
- `em` *em-width*, larghezza della 'M', convenzionalmente circa uguale al corpo del font corrente.

`\fill` è una lunghezza elastica (gomma) di lunghezza naturale nulla ma infinitamente allungabile.

`\stretch{`⟨*moltiplicatore*⟩`}` è una lunghezza elastica (gomma) di lunghezza naturale nulla ma allungabile infinitamente quanto `\fill` moltiplicata per ⟨*moltiplicatore*⟩; questo è un numero decimale con segno facoltativo; se il segno è negativo questa lunghezza diventa 'infinitamente' accorciabile.

`\newlength{`⟨*comando*⟩`}` definisce una nuova lunghezza identificata con ⟨*comando*⟩; questo è il nome di un tipico comando LaTeX formato da una stringa letterale preceduta dal backslash, oppure da un solo segno non letterale preceduto dal backslash. Si possono definire fino a 256 lunghezze; in realtà il motore di composizione eTeX o pdfeTeX accetta la definizione di un numero molto maggiore di lunghezze, ma per compatibilità con

il passato fino ad oggi (2007) non si è ancora visto nessun pacchetto di estensione di una certa importanza farne uso.

`\setlength{`⟨*comando*⟩`}{`⟨*lunghezza*⟩`}` serve per assegnare la ⟨*lunghezza*⟩ esplicita al registro-lunghezza identificato con ⟨*comando*⟩.

`\addtolength`⟨*comando*⟩⟨*lunghezza*⟩ serve per incrementare il valore di lunghezza contenuto nel registro ⟨*comando*⟩ della quantità ⟨*lunghezza*⟩, che può essere sia positiva, sia negativa; in questo caso si ha una sottrazione.

`\settowidth`, `\settoheight` e `\settodepth` seguono la sintassi seguente:

```
\settowidth{⟨comando⟩}{⟨testo⟩}
\settoheight{⟨comando⟩}{⟨testo⟩}
\settodepth{⟨comando⟩}{⟨testo⟩}
```

Ognuna di queste istruzioni assegna al registro-lunghezza ⟨*comando*⟩ rispettivamente la larghezza, l'altezza o la profondità della stringa che costituisce il ⟨*testo*⟩.

E.14.2 Spazi

Gli spazi orizzontali e verticali possono venire inseriti a mano con i seguenti comandi:

```
\hspace{⟨lunghezza⟩}
\hspace*{⟨lunghezza⟩}
\vspace{⟨lunghezza⟩}
\vspace*{⟨lunghezza⟩}
```

I comandi con asterisco impediscono che gli spazi siano eliminati alla fine o all'inizio di una riga o all'inizio o alla fine di una pagina, come invece avviene con gli spazi 'semplici'. Questa possibilità di essere eliminati è essenziale per la buona composizione di una pagina e delle sue righe, ma può non essere quello che si desidera quando si compone dentro ad una scatola oppure quando si eseguono composizioni speciali: si pensi per esempio ad un frontespizio dove non si vuole inserire il primo elemento scritto all'inizio della gabbia, ma lo si desidera spostare in basso per l'equilibrio della pagina.

Molto comodi risultano i comandi abbreviati

```
\bigskip
\medskip
\smallskip
```

che equivalgono rispettivamente a

```
\vspace{\bigskipamount}        \vspace{\medskipamount}
\vspace{\smallskipamount}
```

Le lunghezze indicate con i tre comandi `\...skipamount` sono definiti nei file di classe, ma generalmente essi hanno i valori di 12 pt, 6 pt e 3 pt.

Il comando

`\addvspace{`⟨*lunghezza*⟩`}`

aggiunge spazio verticale tenendo conto di eventuale spazio verticale già inserito implicitamente da comandi precedenti, così che alla fine lo spazio complessivo aggiunto ammonti esattamente alla ⟨*lunghezza*⟩ specificata.

I comandi `\hfill` e `\vfill` equivalgono rispettivamente a `\hspace{\fill}` e `\vspace{\fill}`.

E.14.3 Scatole

Una *scatola* è un oggetto che può contenere diversi segni, anche un testo formato da diverse righe, ma che viene trattato da LaTeX come se fosse un oggetto unico e non lo spezzerà mai alla fine di una riga o di una pagina. I comandi per mettere del ⟨*testo*⟩ dentro una scatola sono

`\mbox{`⟨*testo*⟩`}`
`\makebox[`⟨*larghezza*⟩`][`⟨*posizione*⟩`]{`⟨*testo*⟩`}`
`\fbox{`⟨*testo*⟩`}`
`\framebox[`⟨*larghezza*⟩`][`⟨*posizione*⟩`]{`⟨*testo*⟩`}`

I comandi che cominciano con la lettera 'f' inseriscono una cornice attorno al testo; quelli che cominciano con la lettera 'm' non inseriscono nessuna cornice. Se la ⟨*larghezza*⟩, che è facoltativa, viene specificata, il testo viene inserito dentro una scatola della larghezza specificata, altrimenti i comandi che accettano questo argomento facoltativo si comportano come quelli che non lo richiedono. Se viene specificata la ⟨*larghezza*⟩ allora si può specificare anche la posizione mediante una sola delle lettere seguenti: `l` per collocare il testo a sinistra, `r` per collocare il testo a destra, `s` per distribuire il testo, allargando o restringendo lo spazio interparola, lungo tutta la ⟨*larghezza*⟩; se non viene specificata nessuna posizione il testo risulta centrato dentro la ⟨*larghezza*⟩. si confrontino i casi seguenti.

`\framebox[50mm][l]{testo a sinistra}`	testo a sinistra
`\framebox[50mm][r]{testo a destra}`	testo a destra
`\framebox[50mm]{testo centrato}`	testo centrato
`\framebox[50mm][s]{testo spaziato}`	testo spaziato

Lo spessore della linea che forma la cornice delle scatole è conservata nel parametro dimensionale `\fboxrule` e la distanza della cornice dal testo è conservata dentro il parametro dimensionale `\fboxsep`. Attenzione: lo spessore di 1 pt per la cornice dà luogo a una cornice molto scura: testo. Quindi non è il caso di specificare spessori maggiori se non a ragion veduta.

Con le scatole che accettano la ⟨*larghezza*⟩ se ne può sostituire l'indicazione esplicita mediante quattro nuovi comandi che si riferiscono tutti alla larghezza della scatola composta con `\mbox` che contenga lo stesso ⟨*testo*⟩; esse sono:

`\height` ovvero l'altezza della scatola composta con `\mkbox{`⟨*testo*⟩`}`
`\depth` ovvero la profondità di quella scatola
`\totalheight` ovvero la somma di `\height` e di `\depth`
`\width` ovvero la larghezza di quella scatola.

Per inserire del testo in scatole che contengano materiale 'verticale' (in pratica una successione verticale di righe, uno o più capoversi) si può usare il comando `\parbox` o l'ambiente *minipage*:

`\parbox[`⟨*pos*⟩`][`⟨*altezza*⟩`][`⟨*pos.interna*⟩`]{`⟨*giustezza*⟩`}{`⟨*testo*⟩`}`

oppure

`\begin{minipage}[`⟨*pos*⟩`][`⟨*altezza*⟩`][`⟨*pos.interna*⟩`]{`⟨*giustezza*⟩`}`
⟨*testo*⟩
`\end{minipage}`

dove il parametro ⟨*pos*⟩ indica l'allineamento della scatola con il testo circostante; `t` indica che la riga di base della prima riga della scatola è allineata con la riga di base del testo circostante; `b` indica che la linea di base dell'ultima riga della scatola è allineata con la riga di base del testo circostante; per default, cioè senza specificare nulla, l'asse matematico della scatola è allineato con l'asse matematico del testo circostante. Facoltativamente si può specificare l'⟨*altezza*⟩ della scatola; se questa altezza viene specificata, allora si può usare il terzo parametro facoltativo ⟨*pos.interna*⟩; le lettere di posizionamento sono le stesse, ma si riferiscono a dove il testo contenuto nella scatola viene collocato rispetto ai bordi ideali della scatola stessa. Va infine specificata la ⟨*giustezza*⟩ della scatola, in pratica la sua larghezza e il ⟨*testo*⟩ che essa deve contenere.

La differenza sostanziale fra il comando e l'ambiente è la seguente: il comando `\parbox` non comincia a comporre il ⟨*testo*⟩ al suo interno, finché non l'ha letto tutto; in questo modo il ⟨*testo*⟩ viene elaborato più volte e se contiene comandi fragili questi possono rompersi. Tutto ciò non avviene con l'ambiente *minipage* il quale comincia a comporre il ⟨*testo*⟩ fin dal primo capoverso. In compenso *minipage* in quanto ambiente, richiede una 'amministrazione' un poco più complessa che richiede un maggior tempo di elaborazione. Parlando di microsecondi, non è il caso di formalizzarsi troppo, visti i numerosissimi vantaggi che offre l'ambiente rispetto al comando. L'ambiente *minipage* può anche contenere delle note interne alla 'paginetta' che esso compone; queste note vengono composte con i soliti comandi; solo che la numerazione delle note avviene con lettere corsive in posizione di apice, in modo da distinguerle bene dalle note di piè di pagina, generalmente composte con esponenti numerici o con simboli non letterali.

Si noti che se il contenuto di una *minipage* comincia con una equazione in display, essa conserva il suo spazio che la precede; se non si vuole questo spazio si cominci la *minipage* con `\vspace{-\abovedisplayskip}`.

Il comando `\rule` compone una scatola particolare: essa, infatti, è completamente nera. La sintassi è:

`\rule[`⟨*rialzo*⟩`]{`⟨*base*⟩`}{`⟨*altezza*⟩`}`

La ⟨*base*⟩ e l'⟨*altezza*⟩ sono la base e l'altezza del rettangolo nero; se una delle due è nulla il rettangolo è invisibile, ma l'oggetto continua a mantenere l'altra dimensione; se la base è nulla si ottiene uno *strut*, che in gergo tipografico italiano si chiama *pilastrino*. In questo testo si è mantenuto il nome inglese per ricordare più facilmente i nomi dei comandi che inseriscono *strut* vari sia nelle tabelle, sia in matematica. Il rettangolo nero (o invisibile) risulta rialzato di ⟨*rialzo*⟩ se questa lunghezza è positiva, o ribassato se essa è negativa.

Infine un comando serve per collocare del ⟨*testo*⟩ in una scatola che viene alzata o abbassata a piacere e le si possono assegnare dimensioni a piacere, indipendentemente dal testo che la scatola contiene.

`\raisebox{`⟨*rialzo*⟩`}[`⟨*altezza*⟩`][`⟨*profondità*⟩`]{`⟨*testo*⟩`}`

La ⟨*profondità*⟩ può venire specificata solo se si specifica anche l'⟨*altezza*⟩.

E.15 Disegni e colori

E.15.1 Disegni

Come già specificato l'ambiente *picture* nativo di LaTeX non è in grado di fare altro che semplici disegni, con l'inclinazione dei segmenti e dei vettori molto limitata dagli speciali font che servono per disegnarli; anche i cerchi sia pieni sia vuoti sono un problema, se non nelle situazioni più semplici; i rettangoli ad angoli arrotondati soffrono delle stesse limitazioni dei cerchi; le linee arbitrarie possono essere eseguite solo con le curve di Bézier quadratiche, quindi con certe limitazioni sulle quali qui non è il caso di insistere. Dal 2003 è disponibile il pacchetto **pict2e** che toglie tutte queste limitazioni; esso ha assunto una conformazione stabile dal 2004; quindi si raccomanda caldamente di usare quel pacchetto se si desiderano eseguire semplici disegni.

Se si vuole disegnare qualcosa di più elaborato si ricorra al pacchetto **pgf** e al suo ambiente *tikzpicture*; questo pacchetto consente di disegnare praticamente qualsiasi cosa. Se non dovesse bastare c'è sempre il pacchetto **PSTricks** che è limitato solo dal fatto che il linguaggio PostScript su cui si basa non è onnipotente, ma quasi.

Qui, pertanto, non si darà nessuna informazione sull'ambiente *picture*; gli esempi svolti nel testo dovrebbero essere autoesplicativi. Si invita invece il lettore a documentarsi sui pacchetti **pict2e**, **pgf** e sul suo ambiente *tikzpicture*, nonché sul pacchetto **PSTricks**.

E.15.2 Colori e grafica

Il pacchetto **graphicx** consente di eseguire diverse cose: esso mette a disposizione la gestione del colore, per altro accessibile anche con il solo pacchetto **color** se non si desiderano la altre funzionalità di **graphicx**, e la manipolazione di scatole varie che normalmente non possono essere eseguite da LaTeX, se non passando attraverso il formato PostScript, ma possono venire eseguite da pdfLaTeX. Il lettore è invitato a documentarsi sul file `grfguide.ps` che accompagna il pacchetto; qui si riportano solo i comandi più importanti.

`\scalebox{`⟨*scala-orizz.*⟩`}[`⟨*scala-vert.*⟩`]{`⟨*testo*⟩`}` serve per inserire ⟨*testo*⟩ dentro una scatola per poi scalarla del fattore di scala orizzontale; se il fattore di scala verticale non viene specificato l'ingrandimento verticale avviene con lo stesso fattore di scala orizzontale, altrimenti la scatola viene deformata in maniera diversa così da ottenere effetti particolari.

`\resizebox{`⟨*larghezza*⟩`}{`⟨*altezza*⟩`}{`⟨*testo*⟩`}` esegue in pratica la stessa operazione di `\scalebox` solo che invece di specificare dei fattori di scala, vengono specificate le dimensioni finali effettive. Il comando asteriscato

agisce in modo da scalare l'altezza totale (altezza più profondità) al valore specificato.

`\rotatebox{`⟨*angolo*⟩`}{`⟨*testo*⟩`}` serve per ruotare la scatola che contiene il ⟨*testo*⟩ di un ⟨*angolo*⟩ specificato (in gradi) in senso antiorario.

`\reflectbox{`⟨*text*⟩`}` gira la scatola che contiene il ⟨*testo*⟩ attorno al suo asse verticale in modo che appaia come riflessa allo specchio

`\includegraphics[`⟨*lista di chiavi*⟩`]{`⟨*file grafico*⟩`}` serve per inserire una scatola che contiene l'immagine descritta nel ⟨*file grafico*⟩ elaborata secondo le numerose chiavi che si possono specificare, ognuna con il suo valore; se si specifica il *bounding box*, il rettangolo 'circoscritto', la versione asteriscata di questo comando permette di tagliare tutto quanto sporga fuori da detto rettangolo. Per questo scopo è forse meglio specificare le coordinate del `viewport` o le dimensioni delle strisce da ritagliare con `trim` e poi specificare la chiave `clip`; in questo modo si ha un controllo migliore di quanto si sta eseguendo.

`\definecolor{`⟨*nome*⟩`}{`⟨*modello*⟩`}{`⟨*valore*⟩`}` serve per dare un ⟨*nome*⟩ ad un colore specificato mediante il suo ⟨*modello*⟩ e il valore dei parametri che in quel modello identificano il colore scelto. LaTeX considera come modelli di colore i seguenti:

`gray` è il modello di colore grigio identificato da un solo valore, un numero decimale nell'intervallo 0—1; 0 vuol dire nero e 1 significa bianco; ogni valore intermedio specifica una certa gradazione di grigio.

`rgb` è il modello di colore degli schermi: rosso, verde, blu. I colori sono additivi e il valore che specifica un dato colore in questo modello è dato da tre numeri decimali separati da virgole, tutti e tre nell'intervallo 0—1; 0 vuol dire 'assenza' di quella particolare componente di colore, 1 significa che quella componente di colore è totalmente satura. Per esempio $1, 0, 0$ indica il rosso saturo.

`cmyk` è il modello della quadricromia a stampa con i colori sottrattivi; ogni colore è identificato da quattro componenti corrispondenti ciascuna alle componenti ciano (celeste), magenta (lilla), giallo, nero. Ogni componente è pesata con un numero decimale nell'intervallo 0—1 e la quaterna di valori decimali costituisce una lista di valori separati da virgole.

Con ogni modello di colore, sono predefiniti i seguenti colori: `white`, `black`, `red`, `green`, `blue`, `yellow`, `cyan`, `magenta`. Il driver specifico a seconda se si passi attraverso il programma `dvips` oppure se si proceda con pdfLaTeX, può definire altri colori. I nomi dei colori predefiniti con il driver `dvips` sono contenuti nel file `dvipsnam.def` in `.../tex/latex/graphics/`.

`\color{`⟨*nome*⟩`}` dichiara che da questo momento in poi si userà il colore con il ⟨*nome*⟩ specificato. la sua azione viene delimitata dai gruppi o dagli ambienti.

`\textcolor{`⟨*nome*⟩`}{`⟨*testo*⟩`}` serve per colorare il ⟨*testo*⟩ con il colore denominato ⟨*nome*⟩.

`\colorbox{`⟨*sfondo*⟩`}{`⟨*testo*⟩`}` agisce come `\framebox`, ma il ⟨*testo*⟩ viene composto su uno sfondo colorato con il colore di nome ⟨*sfondo*⟩.

`\fcolorbox{`⟨*bordo*⟩`}{`⟨*sfondo*⟩`}{`⟨*testo*⟩`}` agisce come `\colorbox` salvo che la cornice della scatola non è nera ma colorata con il colore di nome ⟨*bordo*⟩.

`\pagecolor{`⟨*nome*⟩`}` questo comando globale genera uno sfondo con il colore denominato ⟨*nome*⟩ e questo colore permane finché non si dichiara un colore differente (eventualmente `white`).

E.16 Selezione dei caratteri

I comandi standard di LaTeX che hanno a che vedere con la scelta dei font verrà descritta qui di seguito. Per gestire i font in modo più professionale, anzi, per caricare altre collezioni di font e per gestirle al meglio, è necessario documentarsi nella guida `fntguide.dvi`.

E.16.1 Scegliere famiglia, forma e serie

I comandi per scegliere la forma e/o la serie da cui trarre i font per la composizione sono raccolti nella tabella E.4; nella tabella E.5 sono invece appaiate le dichiarazioni e i corrispondenti comandi.

Le dichiarazioni non possono essere usate in matematica; i comandi possono essere usati in matematica, ma non sono soggetti alla variazione automatica di corpo che compete agli indici e ai pedici di primo e di secondo livello, per cui è sconsigliabile farne uso. Piuttosto si ricorra al comando `\text` fornito dal pacchetto **amsmath** e al corrispondente comando `\intertext` per intercalare del testo ad espressioni matematiche incolonnate. Nell'argomento di questi due comandi, se fosse necessario, si può ricorrere tanto ai comandi quanto alle definizioni.

E.16.2 Scegliere il corpo

La dimensione dei caratteri è legata a nomi che fanno riferimento al corpo 'normale', quello, cioè, usato di default; per i corpi inferiori a 10 pt la successione dei corpi procede di 1 pt in 1 pt; per i corpi superiori si procede con una successione geometrica di ragione 1,2; l'opzione di classe *11pt* corrisponde ad un corpo di 10,95 pt, ma il numero magico 10,95 altro non è che $10 \times \sqrt{1{,}2}$. I comandi per la scelta dei corpi relativi a quello normale sono raccolti nella tabella E.6. questi comandi non possono venire usati in matematica.

E.16.3 Corpi testuali e matematici

Si ricorda che fra i comandi disponibili (ma raramente usati) è disponibile il comando per associare i quattro corpi che servono per il testo e gli indici e i pedici di primo e di secondo ordine per la matematica.

`\DeclareMathSizes{`⟨*testo*⟩`}{`⟨*textsize*⟩`}{`⟨*subscriptsize*⟩`}{`⟨*subsubscriptsize*⟩`}`

`\mdseries`	Serie media	`\upshape`	Forma diritta
`\bfseries`	**Serie nera**	`\itshape`	*Forma corsiva*
`\rmfamily`	Tondo con grazie	`\slshape`	*Forma inclinata*
`\sffamily`	Lineare senza grazie	`\scshape`	MAIUSCOLETTO
`\ttfamily`	`Monospaziato`	`\normalfont`	Font di default

Tabella E.4: Dichiarazioni per la scelta di famiglia, serie e forma

`\textmd{`⟨*testo*⟩`}`	`\mdseries`	`\textup{`⟨*testo*⟩`}`	`\upshape`
`\textbf{`⟨*testo*⟩`}`	`\bfseries`	`\textit{`⟨*testo*⟩`}`	`\itshape`
`\textrm{`⟨*testo*⟩`}`	`\rmfamily`	`\textsl{`⟨*testo*⟩`}`	`\slshape`
`\textsf{`⟨*testo*⟩`}`	`\sffamily`	`\textsc{`⟨*testo*⟩`}`	`\scshape`
`\texttt{`⟨*testo*⟩`}`	`\ttfamily`	`\textnormal{`⟨*testo*⟩`}`	`\normalfont`

Tabella E.5: Corrispondenza fra comandi e dichiarazioni

`\tiny`	`\scriptsize`	`\footnotesize`	`\small`	`\normalsize`
`\large`	`\Large`	`\LARGE`	`\huge`	`\Huge`

Tabella E.6: Dichiarazioni di corpo

dove al corpo testuale ⟨*testo*⟩ vengono associati i tre corpi per la matematica: ⟨*textsize*⟩ per i simboli delle espressioni, ⟨*subscriptsize*⟩ per gli indici e i pedici di primo livello e ⟨*subsubscriptsize*⟩ per gli indici e i pedici di secondo livello.

Queste dichiarazioni sono più utili di quanto si pensi, perché talvolta nelle classi standard è desiderabile apportare qualche correzione, specialmente per i corpi più piccoli e più grandi.

E.16.4 Simboli speciali

Qualunque segno di qualunque font può venire stampato prescindendo dalle sue caratteristiche particolari; il comando

`\symbol`⟨*indirizzo*⟩

permette di accedere a qualunque simbolo di qualunque polizza di caratteri mediante il suo ⟨*indirizzo*⟩; il carattere viene stampato prescindendo dalle sue caratteristiche speciali o dall'esistenza di un comando specifico per usarlo; l'unica questione è conoscerne l'indirizzo decimale, oppure ottale, oppure esadecimale. Attenzione: gli indirizzi si riferiscono alla polizza di font usati per la composizione, non ai caratteri introdotti o introducibili con la tastiera; per esempio le due parentesi graffe nell'encoding di entrata hanno indirizzi 123 e 125 rispettivamente: se però si usano questi indirizzi con il font corrente (CM con codifica OT1) si ottiene – e ″, che corrisponde perfettamente con quanto si può leggere nella tabella 11.1 della pagina 127.

Indice analitico